VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Integrace antivirových programů do systému Windows a jejich ochrana vůči odstranění, ochromení či napadení malware

Integrating Anti-Virus Softwares to Windows System and Their Protection Against Removal, Paralysing and Attacks caused by Malware

2019 Bc. Martin Frýdl

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Frýdl**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 1801T064 Informační a komunikační bezpečnost

Téma: Integrace antivirových programů do systému Windows a jejich ochrana vůči odstranění, ochromení či napadení malware
Integrating Anti-Virus Softwares to Windows System and Their Protection Against Removal, Paralysing and Attacks caused by Malware

Jazyk vypracování: čeština

Zásady pro vypracování:

Úkolem je seznámení se s postupy integrace antivirových programů do systému Windows, využití prostředků systému a zabezpečení k jejich ochraně proti ochromení či odstranění škodlivým kódem. Student porovná různé metody nasazení a zabezpečení antivirového programu a navrhne funkční koncept integrace antiviru do systému s ohledem na zabezpečení programu proti ochromení či zneškodnění. Student aplikuje navrhovaný koncept na skutečnou aplikaci. Ta bude obsahovat jak aktivní, tak pasivní ochranné bezpečností prvky, včetně monitoringu změn v systému. Na závěr proběhne testování řešení.

Hlavní body zadání:
1. Seznámení se s postupy vývoje antivirového systému.
2. Seznámení se s architekturou a bezpečnostními mechanismy systému Windows.
3. Návrh vlastního konceptu.
4. Implementace navrženého konceptu s ohledem na aktivní i pasivní zabezpečení.
5. Otestování implementovaného řešení a diskuze výsledků.

Seznam doporučené odborné literatury:

[1] KORET, Joxean a Elias BACHAALANY. The antivirus hacker's handbook. Indianapolis, Indiana: Wiley, 2015. ISBN 978-1-119-02875-8.
[2] SCAMBRAY, Joel. a Stuart. MCCLURE. Hacking exposed Windows: Windows security secrets & solutions. 3rd ed. New York, NY: McGraw-Hill, c2008. ISBN 978-0071494267.
[3] DRÁB, Martin. Jádro systému Windows: kompletní průvodce programátora. Brno: Computer Press, 2011. Programování (Computer Press). ISBN 978-80-251-2731-5.
[4] PERLA, Enrico a Oldani MASSIMILIANO, SPEAKE, Graham, ed. A guide to kernel exploitation: attacking the core. Boston: Syngress, c2011. ISBN 978-1-59749-486-1.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **prof. Ing. Ivan Zelinka, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2019 .....

Rád bych poděkoval prof. Ing. Ivanovi Zelinkovi, Ph.D. za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování diplomové práce věnoval.

Abstrakt

Tato diplomová práce se zabývá integrací antivirových řešení do systému Windows. Jejím hlavním cílem je prozkoumání metodik ochrany antivirových řešení v systému, vytvoření konceptu zabezpečení a implementace tohoto konceptu. Samotný text práce pojednává nejen o teorii s tím spojené, jako je malware, antivirová řešení a jejich integrace, architektura systému Windows a jeho monitorování, ale zejména pak popisuje vytvoření požadovaného konceptu pro zabránění ochromení či odstranění antivirového software.

Klíčová slova: antivirus, malware, Windows

Abstract

This thesis deals with the integration of anti-virus software into Windows. The main goal is to explore the methodologies for protection of anti-virus software in the Windows system and design a security concept and implement this concept. The written content of this thesis describes the related theory such as malware, anti-virus solutions and their integration, Windows architecture and its monitoring and mainly describes the creation of the desired concept to prevent the paralysis or removal of anti-virus software.

Key Words: anti-virus, malware, Windows

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
AV	–	AntiVirus
CD	–	Compact Disc
DLL	–	Dynamic-Link Library
DPC	–	Deferred Procedure Call
DVD	–	Digital Versatile Disc / Digital Video Disc
EFI	–	Extensible Firmware Interface
GUI	–	Graphical User Interface
HAL	–	Hardware Abstraction Layer
HTML	–	HyperText Markup Language
HW	–	Hardware
IM	–	Instant Messaging
MBR	–	Master boot record
MD	–	Message Digest
NT	–	New Technology
OOP	–	Object-Oriented Programming
OS	–	Operating System
PE	–	Portable Executable
SW	–	Software
TPM	–	Trusted Platform Module
UAC	–	User Account Control
UEFI	–	Unified Extensible Firmware Interface
UPX	–	Universal Unpacker
WFP	–	Windows File Protection
WMI	–	Windows Management Instrumentation
WPF	–	Windows Presentation Foundation
WRP	–	Windows Resource Protection

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Antivirové programy (zkráceně též „antiviry“ či AV) jsou software, které slouží k identifikaci a následnému odstranění počítačových virů a jiného škodlivého software (malware). Čtenář bude seznámen se základním rozdělením těchto programů, postupy jejich vývoje a příklady.

Systém Windows je dlouhodobě nejpoužívanějším operačním systémem a většina kybernetických útoků je vedena proti uživateli pracujícím právě na této platformě. Práce nabízí seznámení s architekturou a bezpečnostními mechanismy systému Windows, a to jak z pohledu obránce tak i útočníka.

Silná medializace potřebnosti antivirových řešení a počítačové bezpečnosti má obecně za následek, že většina systémů je dnes zabezpečena alespoň volně šiřitelným antivirovým programem. Tento trend je samozřejmě velmi pozitivní avšak útočníci reagují snahou přímo napadnout tento software za účelem jeho odstranění nebo alespoň ochromení před provedením samotné škodlivé akce v napadeném systému.

Tato práce se zabývá integrací antivirových řešení do systému Windows a využitím prostředků systémů tak, aby bylo zajištěno maximální možné zabezpečení k jejich ochraně proti násilnému ukončení útočníkem.

Práce přichází s vlastním konceptem ochrany antivirových řešení v systému Windows. Čtenář bude seznámen s návrhem tohoto konceptu a jeho implementací. Dosažené výsledky jsou pak zhodnoceny na základě testování hotového řešení.

Samotný text této diplomové práce je členěn do několika kapitol. A je pomyslně rozdělen do dvou na sebe navazujících částí.

Úvodem *teoretické části* definuje kapitola 2 pojem „malware“, popisuje jeho základní typy dle účelu a zmiňuje nejčastější způsoby infikace počítače. V kapitole 3 se čtenář může blíže seznámit s antivirovými systémy, jejich funkcionalitou rozdělenou pomocí základních komponent těchto řešení a způsobem jakým brání napadení chráněného počítače. Dále pak tato kapitola představuje nejznámější komerční antivirové programy a popisuje - porovnává metody jejich integrace do systému Windows. Samotný operační systém Windows je prostřednictvím své architektury popsán v kapitole 4. Na ní navazuje kapitola č. 5 popisující metody a interní nástroje k monitorování systému Windows a jeho prostředků. Kapitola 6 se zabývá bezpečnostními mechanismy implementovanými v systému Windows.

Na základě teoretických poznatků z předešlých kapitol pak vzniká *praktická část*. V kapitole 7 je navrhnut koncept zabezpečení antivirového řešení před násilným ukončením či odstranění tohoto programu. Implementace tohoto konceptu na skutečnou aplikaci je popsána v kapitole 8 a testování hotového řešení pak shrnuje kapitola 9.

2 Malware

Malware je zkratka z anglického „MALicious softWARE“ pro „škodlivý software“, tedy počítačové programy určené k infiltraci nebo poškození počítačů bez souhlasu uživatelů. Malware je obecný pojem zahrnující všechny typy potenciálního nebo přímo škodlivého ohrožení bezpečnosti počítače, jako jsou viry, spyware, červi, trojské koně, rootkity a podobně.[15]

2.1 Rozdělení malware

Existuje široká škála škodlivého software a stále se objevují nové typy a kategorie ohrožení. Malware se dělí do různých kategorií podle mnoha kritérií. Pro účely této práce postačuje rozdělení malware na základní typy dle jeho účelu:[16]

- Počítačový virus
- Červ
- Trojský kůň
- Retrovirus
- Rootkit a bootkit
- Ransomware
- KeyLogger
- Grayware

Tyto základní typy jsou dále krátce popsány.

2.1.1 Počítačový virus

Primární charakteristikou počítačového viru je jeho schopnost se reprodukovat. Tento typ malware bude distribuovat vlastní kopie pomocí jakýchkoli prostředků k šíření.

Viry jsou obsaženy souborech, které musí být spuštěny, aby mohl virus provést škodlivou akci.

Viry lze klasifikovat dle jejich umístění (hostitele):

- Binární spustitelné soubory - zneužití spustitelných souborů (např. .exe., .com, .js), které mohou být spuštěny uživatelem
- Datové soubory - zneužití maker přibalených do datových souborů
- Zaváděcí sektor pevného disku - vir se stane součástí operačního systému v paměti RAM

2.1.2 Červ

Červ pro svou funkcionalitu nepotřebuje žádného hostitele, tedy žádný spustitelný soubor jako je tomu u viru, dokáže se šířit sám bez vědomé asistence uživatele. Využívá počítačových sítí, ve kterých hledá zranitelnosti, které následně využívá ke svému šíření. Velmi rychle tak bývá zpravidla napadena uzavřená síť, např. školní či firemní.

2.1.3 Trojský kůň

Trojský kůň se zpravidla nedokáže replikovat a svévolně se šířit. V dnešní době se však jedná o jednu z nejvíce používaných metod protože tento výraz obsahuje mnoho možností útoků, kde nejsou prováděny tak agresivní akce jako u viru či červa a je proto hůře odhalitelný.

Trojský kůň tak může například sloužit jako: sniffer, keylogger, downloader (stahování dat bez vědomí uživatele), backdoor, spyware aj.

2.1.4 Retrovirus

Na základě biologického termínu „retrovirus“, je počítačový retrovirus takový malware, který aktivně vyhledává antivirový program v počítačovém systému a napadne ho. Retrovir se pokusí zastavit provoz antivirové ochrany. Tyto retroviry nejčastěji obsahují seznam s názvy procesů známých antivrů a tyto procesy pak s použitím dalších technik aktivně vyhledávají a ukončují. Retroviry také bývají označovány jako „anti-antiviry“.

2.1.5 Rootkit a bootkit

Rootkit je druh malware jehož hlavní specifickou vlastností je jeho schopnost maskovat se a skrývat tak svou aktivitu nebo aktivitu jiného malware před bezpečnostními programy. Nežádoucí aktivitu v systému skrývá pomocí maskování: adresářů, v nichž jsou instalovány, položek registru Windows, procesů, síťových spojení a systémových služeb. Mezi jeho základní funkce patří získávání informací, modifikace konfiguračních souborů a spouštění jiných programů. Historicky je za první rootkit považován virus Brain z roku 1986.

2.1.6 Ransomware

Ransomware je typ malware generující přímý zisk tím, že blokuje (nejčastěji zašifrováním) přístup k datům v systému a vyhrožuje, že tyto data zveřejní nebo odstraní pokud nebude vyplaceno výkupné. Toto výkupné je většinou požadováno v kryptoměnách.

Samozřejmě neexistuje žádná záruka, že výplata výkupného vrátí přístup k datům nebo zabrání jejich vymazání. Avšak útočník často skutečně data po obdržení částky zpřístupní za účelem budování si reputace prostřednictvím „spokojených zákazníků“, čímž zvyšuje pravděpodobnost, že i jiné oběti výkupné uhradí.

2.1.7 KeyLogger

Malware zachytává stisky kláves uživatele a to nejčastěji bez jeho vědomí. Keyloggery mohou být využity pro analýzu interakce uživatele s počítačem např. ze strany zaměstnavatele, avšak nejčastěji jsou keyloggery využívány protizákonně ke krádežím přístupových hesel případně dalších citlivých informací, které pak odesílá útočníkovi.

2.1.8 Grayware

Pojem Grayware se používá k popisu nežádoucích aplikací a souborů, jejichž primárním účelem není přímo škodlivá operace, avšak mohou zhoršit výkon počítačů a vést k bezpečnostním rizikům. Tyto programy se chovají spíše nepříjemným nebo obtěžujícím způsobem a v nejhorším případě sledují informace o systému.

Téměř každý komerčně dostupný antivirový software dokáže rozpoznat tyto potenciálně nežádoucí programy.

Grayware zahrnuje především pojmy spyware a adware. **Spyware** slouží pro odesílání dat o uživateli bez jeho vědomí. Nejčastěji se posílají data o kontaktech, přístupu na internet, poloze apod. **Adware** je pravděpodobně jeden z nejvíce lukrativních a nejméně škodlivých typů malware, se specifickým účelem zobrazování reklam na napadeném počítači. Adware obvykle zobrazuje reklamy ve formě vyskakovacích oken, které často nelze zavřít.

2.2 Způsoby infikace malware

Tato kapitola popisuje nejčastější způsoby infikace počítače malware.

2.2.1 Vlastní akce uživatele

Nepozornost a neopatrnost jsou nejčastějším zdrojem nákazy sytému.

Jedná se například o:

- Stažení neověřeného souboru, který může obsahovat škodlivý kód
- Nakažená příloha emailu
- Škodlivý obsah IM zpráv
- Služby umožňující sdílení souborů (např. BitTorrent)

2.2.2 Vyměnitelná média

USB disky, ale také externí pevné disky, CD a DVD mohou obsahovat malware a po připojení do systému počítač infikují. Vzhledem k tomu, že malware může být skrytý ve firmwaru externího média, nikoliv v oblasti úložiště, je velmi obtížné jej odhalit.

2.2.3 Drive-by download

Drive-by download je metoda kdy je škodlivý kód začleněný útočníkem do webové stránky nebo HTML mailu, který se do zařízení oběti stáhne v okamžiku, kdy si ji oběť ve svém prohlížeči zobrazí. Takto stažený kód se pak zpravidla ihned vykoná a to buď velmi destruktivně, nebo častěji pouze na pozadí bez povšimnutí uživatele.

2.2.4 Backdoor

Jedná se o aplikaci na architektuře klient – server. Backdoor slouží pro vzdálený přístup, kdy tímto útočník ovládá daný infikovaný počítač. Po zavedení se připojí na server a čeká na příkazy od útočníka. Tento malware bývá zpravidla speciálně navržen pro napadení jedné konkrétní organizace.

2.2.5 DLL Injekce (Injection)

DLL knihovny[1] mohou být i zneužity za účelem narušení adresního prostoru jiného procesu, injektováním kódem třetí strany. Tento vložený kód je pak zpravidla určen k vykonání jiné, potenciálně nebezpečné akce. Takto napaden může být i samotný antivirový program.

2.2.6 Zero-day útok

„Zero-day attack“ je takový útok, který zneužívá zranitelnosti systému či programu, která není všeobecně známá, a pro kterou ještě neexistuje záplata (patch). Taková zranitelnost může být utajena delší dobu nebo i může zůstat nevyužita, neboť si jí nálezce tzv. „šetří“ za účelem způsobit co nejvíce škod nebo znásobit zisk. Doba, která začne běžet od nalezení zranitelnosti až do uvolnění patche se označuje jako okno zranitelnosti (Vulnerability Windows nebo Window of Vulnerability) a dokud nedojde k nasazení patche, může být této zranitelnosti zneužito.

2.2.7 Sociální inženýrství

Sociální inženýrství zahrnuje všechny sociální techniky s cílem zmanipulovat uživatele tak, aby sám zavedl virus do počítače nebo provedl jinou škodlivou akci za účelem přímého zisku nebo překonání systémové obrany.

Metody sociálního inženýrství mají své kořeny v klasických podvodech reálného světa (vydávání se za příbuzného po telefonu, falešný policista). Jedná se zejména o[11]:

- Vydávání se za někoho jiného - útočník se vydává za někoho jiného za účelem přístupu do systému (např. falešný IT technik) nebo za účelem přímého finančního zisku (např. falešný příbuzný potřebující peníze)

[1] DLL knihovny jsou popsány v kapitole 4.7.

- Vyvolání stresové nebo nebezpečné situace, vyhrožování, nabídka řešení nebezpečné situace, kterou je třeba řešit rychle, vyhnutí se nepříjemnému důsledku
- Vstřícnost - „upřímná“ žádost o pomoc - útočník spoléhá na vstřícnost uživatele pomoci druhému
- Použití lákavé nabídky - výhra v loterii, slevy
- Využití zvědavosti - sexuální obsah

3 Antivirové systémy

Antivirový software je speciální bezpečnostní program jehož účelem je nabídnout lepší ochranu než tu, kterou poskytuje samotný operační systém. Ve většině případů se používá jako preventivní opatření. Tyto systémy jsou navrženy tak, aby sloužily k identifikaci a odstraňování počítačových virů a jiného škodlivého software (malware).

Antiviry používají různé techniky k identifikaci škodlivého softwaru, jako je např. vyhledávání na bázi známých řetězců. Rozvinutější systémy mohou také používat návnady. Kvůli velké škále možných útoku jsou dnes tyto bezpečnostní systémy navrženy tak, aby se zabývaly všemi druhy škodlivých souborů pocházejících jak z důvěryhodných tak i nedůvěryhodných zdrojů.

V naprosté většině případů přichází nejprve útok a následně je proti němu vyvíjena vhodná obrana. S rostoucí sofistikovaností útoků se rozšiřuje perimetr obrany. Ten je potřeba stále vylepšovat a snižovat riziko prolomení obrany malware.

3.1 Funkce antivirových systémů

Moderní antivirové programy nabízejí tyto společné funkce (seznam společných vlastností antivirových produktů)[1]:

- Metody detekce malware
- Schopnost skenovat komprimované a zabalené soubory
- Nástroje pro skenování adresářů na požádání nebo v reálném čase
- Ochrana samotného antivirového systému před napadením
- Firewall a funkce síťové kontroly
- Příkazový řádek a grafické rozhraní
- Konzole pro správu

Tyto základní funkce pak často bývají rozšířené o další možnosti, např. ochrana elektronické pošty (spamový filtr, skenování příloh, apod.), ochrana při prohlížení webových stránek, plánovač úloh, aj.

3.2 Komponenty antivirových systémů

3.2.1 Skenery

Takové nástroje se používají ke skenování kdykoliv se uživatel rozhodne zkontrolovat soubory, obsah adresářů nebo systémovou paměť.

Existují také přístupové (neboli rezidentní) skenery, které se spouštějí automaticky pro analýzu souboru, které jsou vytvořené, upravené nebo spuštěné operačním systémem nebo jinými

programy (jako jsou webové prohlížeče). Tento přístup slouží k zabránění spuštění známých verzí malware.

3.2.2 „Rozbalovače“ (Unpackers)

Jedná se o rutinu nebo soubor rutin vyvinutých pro rozbalení chráněných nebo komprimovaných spustitelných souborů. Malware ve formě spustitelných souborů je běžně zabalen pomocí volně dostupných kompresorů a chráničů nebo vlastních balíčků.

Některé nástroje, jako je UPX, používají jednoduché komprese. Rozbalení souborů komprimovaných pomocí UPX je pak velmi jednoduchá a přímá záležitost. Na druhou stranu existují velmi sofistikované nástroje. Ty transformují kód, který má být zabalen, do bytecode a pak jej pomocí náhodně generovaných virtuálních strojů vkládají do spustitelného souboru, čímž se spustí původní kód malware. Zbavit se této virtualizační vrstvy a odhalit logiku malware je složité a časově náročné.

3.2.3 Databáze vzorků

Antivirový systém uchovává databázi skenovaných vzorků. Tyto vzorky jsou známé části škodlivých souborů. Skener antiviru porovnává skenované soubory oproti této databázi za účelem zjištění zda jsou soubory nebo pakety škodlivé.

Vzorky jsou porovnávány jednoduchými technikami, např. pomocí CRC (kontrolní součty) nebo MD5 hashů.

3.2.4 Emulátory

Emulátor neskenuje soubory, ale spouští je v umělém prostředí, které simuluje skutečný operační systém. Toto prostředí má vlastní virtuální paměť, pevný disk, registr, síť, procesy, atd.

Všechny akce takto spuštěného souboru jsou sledovány, protokolovány pro následnou analýzu a zhodnocení pozorovaného chování malware.

3.2.5 Síťová ochrana

Většina antivirů poskytuje alespoň základní firewall a funkce síťové kontroly, např. paketové filtry, které na základě obsahu hlavičky paketů rozhodují, zda je propustí do systému nebo zda je nepropustí a zahodí (tj. „odfiltruje“).

Od konce devadesátých let až do roku 2010 byl velmi používaný typ škodlivého softwaru tzv. červ[2], který zneužíval jednu nebo více vzdálených zranitelných míst. Antivirové software instalovaly ovladače pro analýzu síťového provozu a detekce nejběžnějších síťových známých útoků. Vzhledem ke snížení podílu tohoto druhu útoku se v mnoha antivirech v současné době jedná o upozaděnou a nepříliš často aktualizovanou ochranu.

[2]Červ je druh malware, který využívá počítačových sítí, kde hledá zranitelnosti, které následně využívá ke svému šíření. Viz kapitola 2.1.2.

3.2.6 Návnady

Většina antivirových systémů obsahuje návnady neboli „honeypoty“. Jejich účelem je na sebe přitáhnou pozornost malware a detekovat napadení. Honeypot je systém připojený, který je nastaven pro zachycení kyberútoků a detekuje, odchyluje nebo studuje pokusy hackerů o získání neautorizovaného přístupu k informačním systémům. Funkce honeypotu je reprezentovat se jako potenciální cíl pro útočníky, shromažďovat informace a informovat obránce o všech pokusech o přístup k honeypotu neoprávněnými uživateli.

3.2.7 Karanténa

Většina antivirových systémů implementuje své řešení karantény umožňující nakažený soubor uzavřít do oddělené části systému kde nemůže způsobit škodu pro jeho pozdější analýzu.

Antiviry často obsahují i sandbox, což je prostředí oddělené od reálného, které umožňuje malware nejen uzavřít, ale také sledovat jeho chování při omezeném přístupu ke zdrojům hostitelského počítače. U některých, často spíše méně komerčních řešení, lze s tímto podsystémem pracovat a malware tak libovolně sledovat a zpracovávat.

3.2.8 Sebeobranné funkce

Mnoho antivirových produktů implementuje techniky pro vlastní ochranu s cílem zabránit nejběžnějším operacím sloužícím útočníkům k ochromení antiviru.

Tato diplomová práce se věnuje právě možnostem sebeobranných funkcí antivirových programů a popisuje tak možný koncept ochrany napříč antivirovými produkty.

3.3 Metody detekce malware

Tato podkapitola popisuje hlavní metody detekce malware používané antivirovými systémy. Tyto metody se kombinují za účelem pokrytí co nejvíce případů a zvýšení úspěšnosti detekce.

3.3.1 Porovnávání vzorků

Metoda „signature-based“ nebo také „pattern-matching“ je základní a nejrozšířenější metoda detekce malware v počítačových systémech. Tato metoda využívá klíčové aspekty zkoumaného souboru k vytvoření otisku známého malware. Tento záznam pak může představovat řadu bajtů v souboru nebo hash souboru či hash jeho části (nejčastěji začátek a konec souboru).

Velkým omezením této metody detekce založené na podpisu je, že plně závisí na velikosti a aktuálnosti databáze poskytující vzorky k porovnání a není sama schopna označit škodlivé soubory, jejichž vzorky nebyly nikdy zachyceny. S tímto vědomím se tak útočníci často snaží mutovat jejich výtvory změnou kódu malware čímž se změní struktura souboru a nebude při skenování odhalen.

3.3.2 Chování malware

Pozorování chování vzorku neboli „Behavior-based“ metoda nezkoumá kód malware, ale jeho chování v systému. Při použití této metody jsou shromažďovány programy se stejným chováním. Jediný odhalený vzor škodlivého chování může pomoci identifikovat různé vzorky malware.

Velkou výhodou použití tohoto systému je jeho odolnost vůči mutacím malware, které zásadně mění jeho kód nikoli však jeho chování. Nevýhodou této metody je pak fakt, že z logiky věci detekuje malware pouze v době, kdy provádí svou činnost.

3.3.3 Sandbox prostředí

Sandbox je virtuální prostředí oddělené od reálného, které umožňuje malware nejen uzavřít ale také sledovat jeho chování při omezeném přístupu ke zdrojům hostitelského počítače.

Tato detekční metoda je podobná detekční metodě založené na chování. Antivirus spouští programy v tomto virtuálním prostředí a zaznamenává, jakou akci provádí program namísto detekce při běhu v reálném systému. Identifikace škodlivého programu antivirovým softwarem je umožněna ověřením aktivit programu. Pokud je program určen jako neškodný je následně spuštěn v reálném prostředí.[13]

Tato technika je, vzhledem k její náročnosti, používána v antivirových řešeních určených k potřebě koncového uživatele jen velmi zřídka.

3.3.4 Heuristické metody

Jak zmiňují kapitoly 3.3.1 a 3.3.2, metody založené na porovnávání kódu či chování vzorků mají některé nevýhody. Heuristické metody detekce malware jsou navrženy k překonání těchto nevýhod a pro detekci používají techniky dolování dat („data mining“) a strojového učení („machine learning“).[12] Pojem „heuristická analýza“ obsahuje vícero různých technik.

Metoda známá jako **statická heuristická analýza** zahrnuje dekompilaci podezřelého programu a zkoumání jeho zdrojového kódu. Tento kód je pak porovnán s viry, které jsou již známé a jsou v databázi. Pokud určité procento zdrojového kódu odpovídá čemukoli v heuristické databázi, kód je označen jako možná hrozba. Antivirový nástroj hledá pomocí heuristických metod přítomnost vzácných instrukcí nebo podezřelých částí kódu ve skenovaném souboru. Jediný podezřelý atribut nemusí být dostatečným argumentem pro to, aby byl soubor označen jako škodlivý. Nicméně několik takových charakteristik může překročit očekávanou rizikovou hranici, což vede k tomu, že nástroj klasifikuje soubor jako malware.[14]

Dynamická heuristika pak izoluje podezřelý program nebo kus kódu uvnitř specializovaného virtuálního stroje nebo karantény a poskytuje antivirovému programu šanci otestovat kód a simulovat, co se stane, kdyby byl podezřelý soubor spuštěn. Zkoumá každý příkaz a hledá podezřelé chování, jako je například samoreprodukce, přepsání souborů a další akce, které jsou pro viry běžné.

Heuristická analýza je ideální pro identifikaci nových hrozeb. Největší nevýhodou heuristiky je, že může neúmyslně označit legitimní soubory jako škodlivé. Z tohoto důvodu jsou často heuristické nástroje typicky jen jedna z detekčních metod v sofistikovaném antivirovém systému, kde je nasazena společně s dalšími metodami detekce virů, jako je analýza vzorků a další proaktivní technologie.[17]

3.4 Zranitelnosti antivirových řešení

Obecné povědomí o kyberkriminalitě a ohrožení malware stále stoupá a s tím i počet zařízení chráněných některým z antivirových řešení. Většina uživatelů si myslí, že samotný antivirus ochrání jejich počítač před většinou možných ohrožení. Právě tato, často až absolutní, víra v používaný antivirus z něj dělá perfektní cíl. Existují útoky vedené přímo proti antiviru za účelem jeho ochromení a poté dalšího pokračovaní útoku. Antivirus tedy musí chránit i sám sebe a uživatel by se měl zajímat o vnější vlivy, které mohou působit na antivirus.

Tvůrci malwaru a virů vždy přicházejí s novými způsoby, jak získat přístup k vašemu počítači a dalším zařízením podporujícím internet (včetně mobilního telefonu a tabletu). Bezpečnostní společnosti neustále aktualizují své antiviry. **Největší zranitelností** antiviru (potažmo i systému) **je** právě samotný **neaktualizovaný antivirus**.

Mnoho antivirových produktů implementuje techniky vlastní ochrany v jádrech ovladačů s cílem zabránit nejběžnějším operacím, které by je mohly ohrozit.

Na trhu se nachází řada antivirových produktů, proto je nemožné cílit na všechny uživatele a jejich antiviry stejnou technikou. Útočníci používají exploity zaměřené na méně různorodý, ale mnohem populárnější software, jako jsou webové prohlížeče (Firefox, Chrome a další) nebo kancelářské produkty (Microsoft Office, OpenOffice, apod.).[1] Dále útočník zvolí techniku podle typu uživatele. Tato práce se zaměřuje zejména na „domácí uživatele“. Útočníci chtějí maximalizovat počet infikovaných uživatelů, a proto používají spíše jednodušší techniky, pomocí kterých dosahují rychlých výsledků. Malé až středně velké společnosti jsou v mnoha směrech velmi podobné domácím uživatelům. Útočníci zaměřující se na menší společnosti by pravděpodobně používali techniky podobné těm, které byly použity k útoku na domácí uživatele, jako např. sociální inženýrství či využití chyb v neaktualizovaných programech. Je velmi nepravděpodobné že by útočník využil zero-day útok[3] k útoku na menší firmu či „domácího“ uživatele, připravil by se tím o možnost většího profitu. Zajímavějšími cíli jsou vlády a velké firmy. Útok na ně vyžaduje použití složitějších technik.

3.5 Známé antivirové systémy

V této kapitole je uvedeno několik nejznámějších (nejpoužívanějších) antivirových systémů resp. jejich výrobců. Na základě zadání této práce se budeme zabývat pouze software, který je určený

[3]Zero-day útok využívá nepublikované chyby v zabezpečení softwaru. Více v kapitole o klasifikaci malware kapitola 2.2.6.

po operační systém Windows.

3.5.1 Avast

Avast[4] antivirus vyvíjí česká nadnárodní společnost Avast Software s.r.o. založená roku 2010. V současnosti tento anti-malware produkt zaujímá největší podíl na trhu s antivirovými software (viz obrázek 1). Je velmi oblíben uživateli pro svou bezplatnou verzi a jednoduchost.

3.5.2 Rodina antivirových řešení Eset

Rodina antivirových řešení od společnosti Eset[5] poskytuje (dle názoru autora této práce a pravidelnému oceňování těchto antivirů v nezávislých testech) účinné a rychlé řešení s minimálními nároky na systém. Dále tato práce bude pojednávat o Antiviru Nod32 z této rodiny.

3.5.3 Malwarebytes

Software společnosti Malwarebytes Corporation[6] vyvíjí svůj antivirus od roku 2006. Je dostupný i v bezplatné verzi. Díky využití pokročilých technologií dokáže skener Malwarebytes odolat snaze o blokování ze strany malware, který se tím pokouší zabránit svému odstranění.

3.5.4 McAfee

McAfee, Inc., původně McAfee Security[7] je americká počítačová firma produkující bezpečnostní software. Její produkt McAfee antivirus je jedním z nejpoužívanějších. V porovnání s jinými antiviry je jeho administrace o něco složitější a poněkud těžkopádnější.

3.5.5 Bitdefender

Společnost Bitdefender[8] byla založena roku 2001. Jejich antivirové řešení se kromě jiného zaměřuje na bezpečnost elektronické pošty. Antivir svojí činností jen minimálně zatěžuje počítač a navíc obsahuje správce hesel pro rychlé a bezpečné přihlašování k internetovým službám a profily nastavení respektují uživatelovu aktuální činnost s počítačem.

3.5.6 Webroot

Společnost Webroot[9] se během posledních let stala jednou z významných firem poskytujících bezpečnostní řešení. Jejich antivirus je založen na cloudových technologiích. Tato aplikace obsahuje funkci Social Network Protection (ochrana sociální sítě), která umožňuje ochranu na Twitteru a Facebooku.

[4] https://www.avast.com
[5] https://www.eset.com
[6] https://www.malwarebytes.com
[7] https://www.mcafee.com
[8] https://www.bitdefender.com
[9] https://www.webroot.com

3.5.7 Avira

Avira Antivirus[10] je starší antivirový program vyvíjený od roku 1988 německou firmou Avira. Tento bezpečnostní program je nejen dobře ovladatelný a nenáročný na systémové zdroje. Ověřuje každý otevřený a zavřený soubor. Dále umí detekovat a mazat rootkity.

3.5.8 Windows Defender

Windows Defender (původním jménem Microsoft AntiSpyware) je integrován v OS Windows a spouští se automaticky. Jedná se o jakousi základní ochranu před malware přímo od společnosti Microsoft, která se neustále vyvíjí a zlepšuje a dnes je již téměř srovnatelná s nejlepšími antiviry na trhu.

Produkt mimo samočinné vyhledávání hrozeb poskytuje prostřednictvím tzv. „Centra akcí“ i nástroje na monitorování některých částí systému. V závislosti na verzi se jedná např. o: *Firewall a ochrana sítě*, *Řízení aplikací a prohlížečů*, *Nástroje rodičovské kontroly*, *Výkon a stav zařízení*, aj.

Stejně jako většina antivirových programů prohledává systém a monitoruje chování software. Je-li připojena jednotka USB aplikace Defender automaticky skenuje její obsah. Nové podezřelé položky jsou nahrány na cloud k analýze, což je funkcionalita, kterou lze zakázat zrušením automatického předávání vzorků v centru zabezpečení systému Windows.

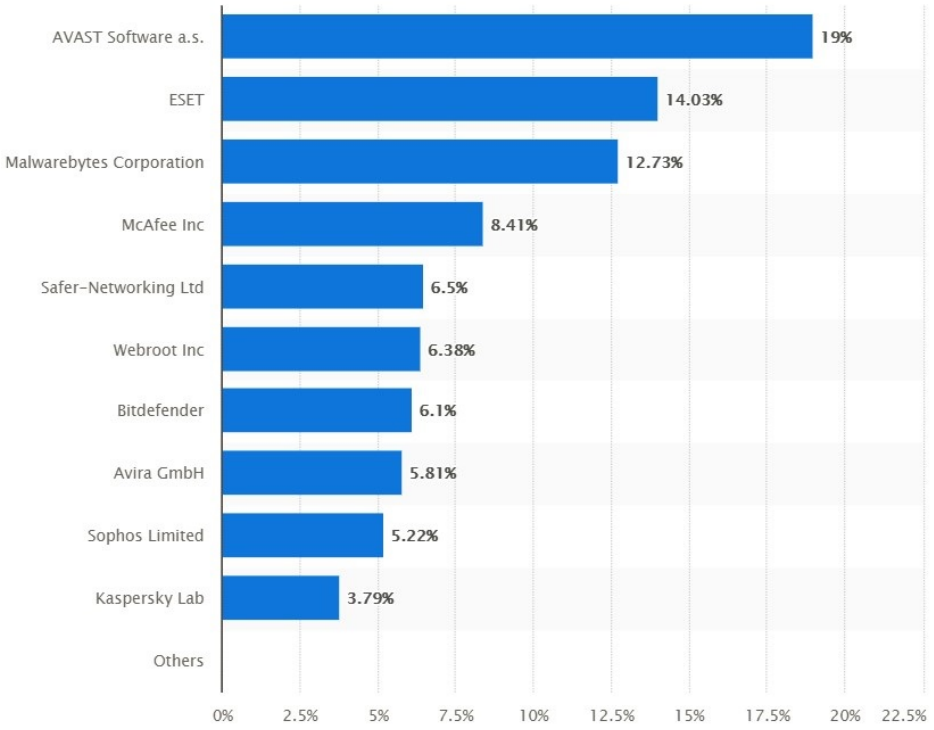
Do Defenderu bylo od poloviny roku 2017 přidáno několik dalších nástrojů a technik. Nová specializovaná obrana proti ransomware řídí přístup ke klíčovým složkám, aby se zabránilo přepisování dat a zvýšila se funkce ochrany sítě předcházející jejich odcizení ze systému. S aktualizací pro verzi systému 10 může Windows Defender pracovat v sandboxu, což přidává další silnou ochrannou vrstvu. Vhod přijde v případě, že se malwaru podaří antivir kompromitovat.

3.5.9 Podíl antivirových řešení na trhu

Obrázek 1 reprezentuje graf[2] poskytující přehled o rozdělení trhu s antivirovými systémy pro první pololetí roku 2018 (cit. 11.7.2018) a OS Windows. Můžeme pozorovat, že dle tohoto grafu je mezi uživateli operačního systému Windows nejpoužívanějším antivirovým řešením[2] software Avast (19%).

[10] https://www.avira.com

Obrázek 1: Procentuální rozdělení trhu s antivirovými systémy pro první pololetí roku 2018 a OS Windows



3.6 Přenositelné antivirové programy

Níže je uveden seznam[10] přenositelných (portable) antivirových řešení nevyžadujících instalaci do systému. Přenositelné antivirové programy tak mohou být spouštěny z lokálního adresáře, externích disků, optických medií nebo i z cloudových úložišť.

3.6.1 ClamWin Portable

ClamWin[11] je bezplatný antivirus, který poskytuje vysokou míru detekce virů a spyware a pravidelné aktualizace antivirové databáze. Program ClamWin umožňuje ruční prohledávání jednotlivých souborů, složek nebo disků, ale neobsahuje skener fungující na pozadí systému.

3.6.2 Emsisoft Emergency Kit Portable

Emsisoft[12] Emergency Kit obsahuje výkonný skener Emsisoft s grafickým uživatelským rozhraním. Dle výrobce dokáže identifikovat všechny druhy malware v napadeném počítači.

[11] http://www.clamwin.com
[12] https://www.emsisoft.com/en/software/eek/

3.6.3 HijackThis Portable

HijackThis[13] prověřuje nastavení prohlížeče a operačního systému počítače a generuje záznam o aktuálním stavu, na jehož základě lze volitelně odstranit nežádoucí nastavení a soubory z počítače.

3.6.4 Kaspersky TDSSKiller Portable

Nástroj Kaspersky TDSSKiller[14] slouží k odhalování bootkitů a rootkitů[15]. Nejedná se o plnohodnotný antivirový nástroj, může však být užitečný při odstraňování rootkitů z infikovaných počítačů se zastaralou nebo žádnou nainstalovanou ochranou.

3.6.5 McAfee Stinger Portable

U řešení Stinger[16] společnosti McAfee je kladen velký důraz na odhalování „falešných poplachů“. Nejedná se o náhradu za kompletní antivirový program, ale v mnoha situacích je vhodný a lze použít i jako doplnění k instalovanému řešení.

3.6.6 Spybot - Search & Destroy Portable

Spybot - Search & Destroy[17] detekuje a odstraňuje spyware[18]. Může také vymazat „stopy“, záznamy využití počítače, což je zajímavá funkce pro sdílené zařízení. Umožňuje také opravit některé nekonzistentní záznamy v registrech.

3.7 Integrace antivirových řešení do systémů Windows

Pro svou funkcionalitu musí být anivirový program do systému Windows integrován tak, aby dokázal pracovat s jádrem OS a mohl monitorovat systémové události.

3.7.1 Ovladače

V současné době je většina antivirových řešení dostupných na trhu a integrována do systému pomocí ovladačů, které si samy vytvářejí a jsou instalovány spolu s instalací AV.

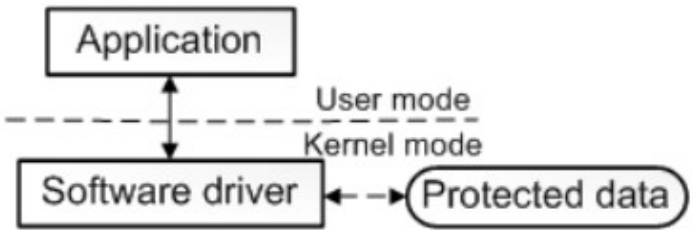
Obecně chápeme ovladače jako „prostředníka“ mezi systémem a HW zařízením. V případě antivirových systémů a dalších programů však ovladače nejsou spojeny s žádným hardwarovým

[13] https://portableapps.com/apps/security/hijackthis-portable
[14] https://usa.kaspersky.com/downloads/tdsskiller
[15] Jako bootkit či rootkit se klasifikuje malware jehož hlavní činností je maskování škodlivé činnosti před ochranou počítače. Více v kapitole 2.1.5.
[16] https://www.mcafee.com/enterprise/en-in/downloads/free-tools/stinger.html
[17] https://portableapps.com/apps/security/spybot_portable
[18] Spyware slouží zejména pro sledování aktivit uživatele napadeného počítače. Více v kapitole o klasifikaci malware 2.1.8.

zařízením. Antivirový software potřebuje přístup k datovým strukturám operačního systému, ke kterým lze přistupovat pouze kódem spuštěným v režimu jádra.[19]

Přístupu k jádru operačního systému lze docílit vytvořením softwarového ovladače. Vyvíjený nástroj je rozdělen na dvě složky. První komponenta běží v uživatelském režimu a představuje uživatelské rozhraní. Druhá komponenta běží v režimu jádra a má přístup k základním datům operačního systému. Součást, která je spuštěna v uživatelském režimu, se nazývá aplikace a součást, která je spuštěna v režimu jádra, se nazývá softwarový ovladač. Obrázek 2 znázorňuje aplikaci běžící v uživatelském režimu komunikující s ovladačem softwaru v režimu jádra.

Obrázek 2: Aplikace v uživatelském režimu komunikující s ovladačem softwaru[19]



Softwarové ovladače běží vždy v režimu jádra. Hlavním důvodem je získání přístupu k chráněným datům, která jsou dostupná pouze v tomto režimu.

3.7.2 Monitorování volání Windows API

Antivirový systém musí sledovat události v systému. K tomuto se nejčastěji využívá metoda „Windows API hooking“[20].

Hooking API systému Windows je proces umožňující zachytit volání funkcí z tohoto API, a tím umožňuje kontrolu nad tím, jak se chová operační systém nebo software.

Hooky na Windows API lze rozdělit do následujících typů:[20]

- Místní - ovlivňují pouze specifické aplikace,
- Globální - týkají se všech systémových procesů.

Předpokládá se, že antivirové systémy využívají oba typy se zaměřením zejména na globální pro odchytávání velkého množství systémových událostí.

Například infrastruktura *AppInit_DLLs* načte předdefinovanou sadu knihoven DLL do všech procesů uživatelského režimu spojených s knihovnou User32.dll (ve skutečnosti nejsou téměř žádné spustitelné soubory, které by s ní nebyly propojeny). Svými metodami pak ovlivňuje

[19] Více o rezdělení systému na uživatelský prostor a prostor jádra v kapitole 4.8.
[20] Více o Windows API v kapitole 4.6.

téměř všechny spustitelné procesy v systému, na rozdíl například od metody *SetWindowsHooks*, která je omezena pouze na vybranou plochu[20].

3.8 Nasazení antivirových řešení

3.8.1 Instalace antivirových řešení

Naprostou většinou dnes používaných instancí jednotlivých antivirů jsou instalovaná řešení, kdy si uživatel bezplatně či za úplatu stáhne (nejlépe pouze z oficiálních stránek výrobce) hotový program, který následně nainstaluje do svého systému. Přičemž je doporučeno úplné odstranění všech stop předchozí AV klienta, jelikož používání dvou odlišných řešení, které soutěží o systémové zdroje, by mohlo způsobovat konflikty v průběhu skenování v reálném čase a zhoršenému a nepravidelnému výkonu systému.

3.8.2 Online antiviry

Online antivirová řešení nabízí většina větších bezpečnostních firem např Kaspersky či Eset. Odeslání celého obsahu disku na vzdálený server ke kontrole by bylo samozřejmě velmi nepraktické, tyto antiviry tak pro své spuštění v prohlížeči požadují stáhnutí výkonného jádra a antivirové databáze. Pro tuto práci jsou to tak soubory vyžadující ochranu proti ochromení či odstranění.

3.8.3 Porovnání nasazení antivirových řešení

Ačkoli může mít využití přenositelných nebo online antivirových řešení v některých specifických situacích své opodstatnění, nejedná se zpravidla o plnohodnotné antivirové systémy určené ke komplexní ochraně systému. Jejich výhoda spočívá zejména v rychlosti a znovupoužitelnosti na vícero zařízeních. Hlavní nevýhodou je pak omezenost funkcí, jako absence zabezpečení elektronické pošty, truhla sloužící k uchovávání malware, u online řešení pak často absence jakékoli funkce pro odstranění malware, apod. Instalované řešení je lépe integrováno do systému a může tak interagovat s OS a zachytávat jeho události.

Automaticky integrovaný program Windows Defender je dnes už na velmi dobré úrovni a lze ho považovat za dostatečné bezpečnostní řešení. Použití Defenderu je vhodné na úkor bezplatných verzí antivirových řešení, které většinou nedosahují takových kvalit.

Bezpečnost systému bychom tak měli zajistit spíše použitím komplexního instalovaného antivirového systému. A přenositelná řešení pak využít spíše jako doplnění hlavního antivirového software v případě potřeby některé specifické funkcionality.

4 Architektura operačního systému Windows

4.1 Architektura Windows NT

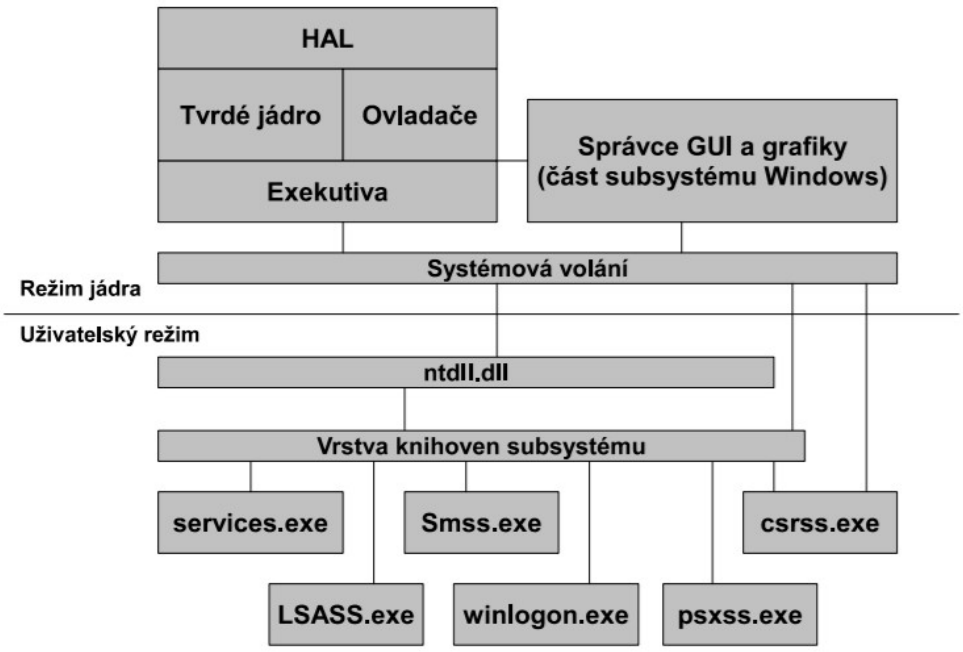
Pojmem Windows označujeme rodinu operačních systémů od firmy Microsoft. Jedná se o nejčastěji používaný OS na světě a pro běžného uživatele prakticky jediný myslitelný.

Windows NT je označení pro řady operačních systémů firmy Microsoft, které jsou určeny pro osobní počítače. Systém Windows vychází kromě desktopových verzí také ve verzích mobilních a serverových. Tato práce se dále zaměřuje na momentálně masově nejpoužívanější a nejaktuálnější desktopový operační systém Windows 10 (řada NT 10.0).

Systémy Windows NT patří do skupiny monolitických operačních systémů, které na rozdíl od mikrojádrových (mikrokernel) systémů mají velké jádro obsahující naprostou většinu podsystémů nutných pro běh celého systému. Tyto podsystémy (např. souborový systém, správa procesů, bezpečnostní model aj.) sdílejí společný virtuální adresový prostor, což vede ke zrychlení jejich vzájemné komunikace a tedy většímu výkonu systému, avšak za cenu nižší stability a bezpečnosti.

Systémy Windows NT pro svou činnost využívají běhu několika málo procesů zajišťujících správu služeb, či autentizaci. Většina funkcí pak běží v režimu jádra, které je téměř celé naprogramováno v jazyce C.

Obrázek 3: Architektura Windows NT[4]



Na obrázku 3 lze vidět obecnou architekturu systémů Windows NT. Některé zobrazené komponenty dále popisují následující odstavce.

4.2 Vrstva abstrakce hardwaru (HAL)

HAL se v architektuře nachází na nejnižší úrovni a slouží, jak název napovídá, k potlačení specifik HW, jako například různé modely procesů a napomáhá tak přenositelnosti aplikací mezi různými HW.

Programátor se tak při psaní běžných aplikací a ovladačů nemusí zabývat odlišnostmi, které mohou způsobit rozdílné HW modely.

4.3 Tvrdé jádro

Tvrdé jádro je tenká vrstva nad HAL implementující jednoduché mechanismy, např. obsluha systémových volání, odložené volání procedur - DPC, plánování vláken procesoru, využívané vyššími vrstvami pro stavbu složitějších struktur.

4.4 Exekutiva

Využívá služeb tvrdého jádra k realizaci mnohem složitějších procedur. Exekutiva se skládá z několika součástí, kde každá tato komponenta poskytuje sadu rutin, které mohou volat libovolný kód běžící v režimu jádra. Komponenty spolu navzájem komunikují pomocí přesně definovaných rozhraní.

Jednotlivými komponentami exekutivy jsou:

- **Správce objektů (Object Manager)** - zapouzdřuje principy platící pro všechny druhy objektů a obsahuje jednotný mechanizmus řízení přístupu. Psaní kódu pak programátorovi může připomínat principy běžně známé v OOP.
- **Správce paměti (Memory Manager)** - zaštiťuje činnosti související se správou fyzické či virtuální paměti např. mapování adres, přidávání či odebírání paměťového prostoru atd.
- **Správce vstupně/výstupních zařízení (I/O Manager)** - zajišťuje funkce související s ovladači, jako načítání a uvolňování ovladačů v paměťovém prostoru jádra za běhu nebo komunikace mezi ovladači.
- **Správce procesů (Process Manager)** - se stará o spouštění, běh a ukončování procesů a vláken. Poskytuje ostatním komponentám informace o právě bežících procesech, možnost měnit jejich prioritu nebo je i násilně ukončit.

4.5 Subsystémy

Subsystémy rozumíme prostředí pro aplikace. V systému Windows se nacházejí dva subsystémy, a sice POSIX a Windows.

Subsystém POSIX označuje sadu mezinárodních standardů popisujících aplikační rozhraní v operačních systémech založených na Unixu. POSIX se spouští, pokud uživatel spustí proces určený pro tento subsystém.

Subsystém Windows je na rozdíl od POSIX spuštěn neustále, neboť by bez něj systém nemohl pracovat. Jeho hlavním úkolem je vykreslování oken konzolových aplikací. Je informován o běžících procesech a vláknech, přičemž si uchovává vlastní kopii těchto seznamů aby pro ně mohl vykonávat požadavky při volání funkcí z Windows API.

4.6 Windows API (Windows Application Programming Interface)

Windows API umožňuje aplikacím využívat možnosti OS Windows. Pomocí tohoto API lze vyvíjet aplikace spustitelné ve všech verzích systému Windows a současně využívají funkce a možnosti jedinečné pro každou verzi, neboť se stále vyvíjí. (Dříve se toto rozhraní nazývalo Win32 API. Název API Windows přesněji odráží jeho kořeny v 16bitovém systému Windows a jeho podporu na 64bitových systémech.)[9]

Windows API umožňuje přístup k primárním zdrojům operačního systému Windows, jako jsou soubory, procesy, vlákna nebo registry. Nabízí bezpečností funkce, které zajišťují autentizaci, autorizaci, kryptografii a další podobné služby. Dále také grafické funkce, které umožňují vytvářet grafiku na monitoru, tiskárně a dalších výstupních zařízeních. Funkce uživatelského rozhraní vytvářejí okna, tlačítka, boxy a různé další grafické komponenty. Multimediální subsystém umožňuje vytvářet videa, zvuk a práci se vstupními zařízeními. A mnoho dalších funkcí.

Možnosti Windows API lze dle funkcionality rozdělit do následujících kategorií:

- Administrace a správa
- Diagnostické nástroje
- Grafika
- Multimédia
- Síťové služby
- Bezpečnost
- Systémové služby
- Uživatelské rozhraní

4.7 Dynamicky linkované knihovny (DLL)

V operačních systémech Microsoft Windows je většina funkcí operačního systému poskytována pomocí DLL souborů. DLL je knihovna, která obsahuje kód a data, které může používat více

programů najednou (použitím DLL knihoven vznikají závislosti). Windows Comdlg32 DLL provádí běžné funkce související s dialogovým oknem. Proto každý program může použít funkci dialogového okna „Otevřít“[3], která je obsažena v této knihovně DLL.

Některé programy mohou obsahovat mnoho různých modulů a každý modul programu je obsažen a distribuován v knihovnách DLL. Většinu funkcí takového programu může poskytnout právě DLL knihovna.

Použití knihoven DLL pomáhá podporovat modularizací kódu, opakované použití kódu, usnadnění aktualizací a efektivní využití paměti. Proto se operační systém a programy načítají rychleji, běží rychleji a na počítači zabírají méně místa na disku.

4.8 Uživatelský prostor a prostor jádra

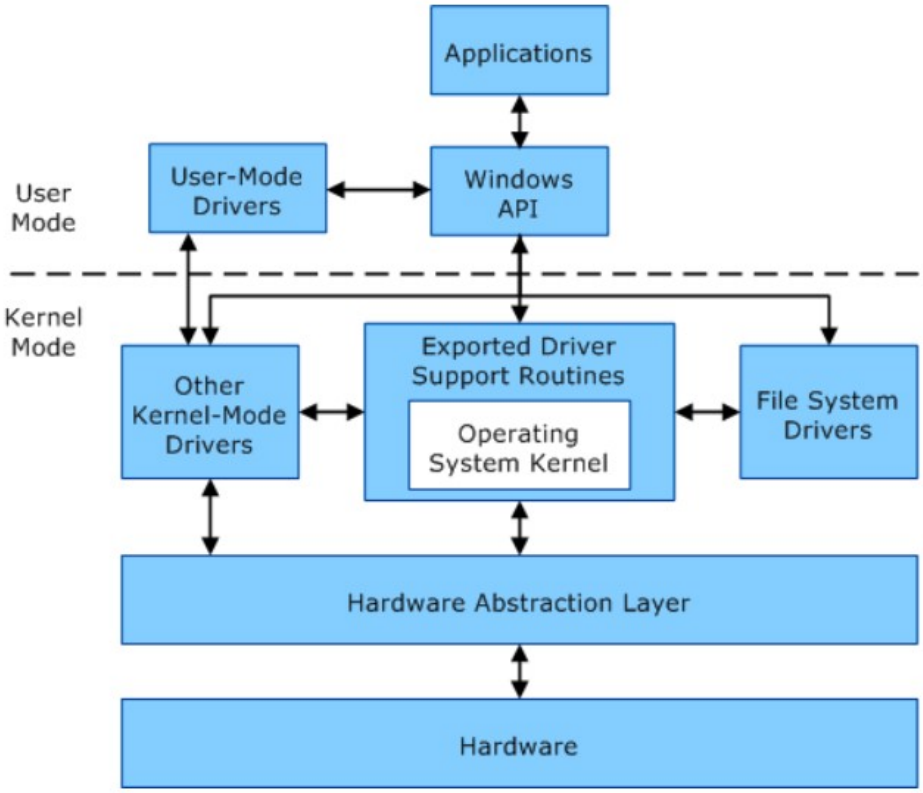
Uživatelský prostor a prostor jádra (často nepřekládáno z „User space“ a „Kernel space“) je označení pro dvě oddělené části virtuální paměti. Procesor v zařízení systémem Windows pracuje ve dvou režimech: uživatelský režim a režim jádra. Procesor přepíná mezi oběma režimy v závislosti na typu kódu. Aplikace běží v uživatelském režimu a základní komponenty operačního systému jsou spuštěny v režimu jádra.

Když se spustí aplikace uživatelského režimu, systém Windows vytvoří proces aplikace. Tento proces poskytuje aplikaci soukromý virtuální adresový prostor, který je přidělován na úrovni procesu a všechna vlákna jednoho procesu spolu tyto adresy sdílí, tudíž při vícevláknovém programování je potřeba předcházet konfliktům mezi vlákny. Vzhledem k tomu, že virtuální adresní prostor aplikace je soukromý, jedna aplikace nemůže měnit data, která patří jiné aplikaci. Každá aplikace běží izolovaně a pokud dojde k selhání aplikace, havárie je omezena pouze na tuto jednu aplikaci. Jiné aplikace a operační systém nejsou havárií ovlivněny. Proces běžící v uživatelském režimu nemůže získat přístup k virtuálním adresám, které jsou vyhrazeny pro operační systém. Omezení virtuálního adresového prostoru uživatelské aplikace zabraňuje aplikaci, aby měnila a případně poškozovala kritická data operačního systému.

Prostor jádra je určen procesům operačního systému jako správa paměti, většina ovladačů apod. Veškerý kód, který běží v režimu jádra, sdílí jediný virtuální adresní prostor. To znamená, že ovladač režimu jádra není izolován od jiných ovladačů a samotného operačního systému.Pokud selže ovladač režimu jádra, dojde k selhání celého operačního systému.

Obrázek 4 znázorňuje komunikaci mezi komponentami pracujícími v režimu jádra a v uživatelském režimu.

Obrázek 4: Komunikace mezi komponentami pracujícími v režimu jádra a v uživatelském režimu[8]



5 Monitorování systému Windows

Monitorování je jedním z nejdůležitějších předpokladů bezpečného systému. Sledováním systémového provozu jsme schopni odhalit mnoho potenciálních nebezpečí. Tato kapitola popisuje možnosti monitorování systémových prostředků v OS windows.

5.1 Procesy

Při každém spuštění programu v systému Windows dojde k vytvoření jednoho nebo i vícero procesů. Každému procesu je přidělen vlastní adresový prostor. Proces tak může ovlivňovat pouze vlastní paměť a tak žádná jeho činnost nemůže narušit běh samotného systému nebo ostatních procesů.

Určitý proces (kód) může být spuštěn i bez nutnosti mít fyzicky uložený soubor na disku. Monitorování spuštěných procesů je základním kamenem optimalizace výkonu a bezpečnosti systému.

Systém Windows standardně obsahuje velmi známý program „Správce úloh“ neboli „Task Manager“, který je možno použít mimo jiné i právě ke sledování spuštěných procesů. Jedná se však o uzavřené řešení bez možnosti úpravy.

5.2 Paměť RAM

Sledováním využití paměti RAM můžeme nejen pozorovat využívání tohoto systémového prostředku daným procesem, ale také lze odhalit DLL Injection útok, který může způsobit nárůst využívané paměti daným procesem. Pro monitorování využití paměti RAM jednotlivými procesy existuje velké množství externích programů a základní informace poskytuje také nástroj „Správce úloh“.

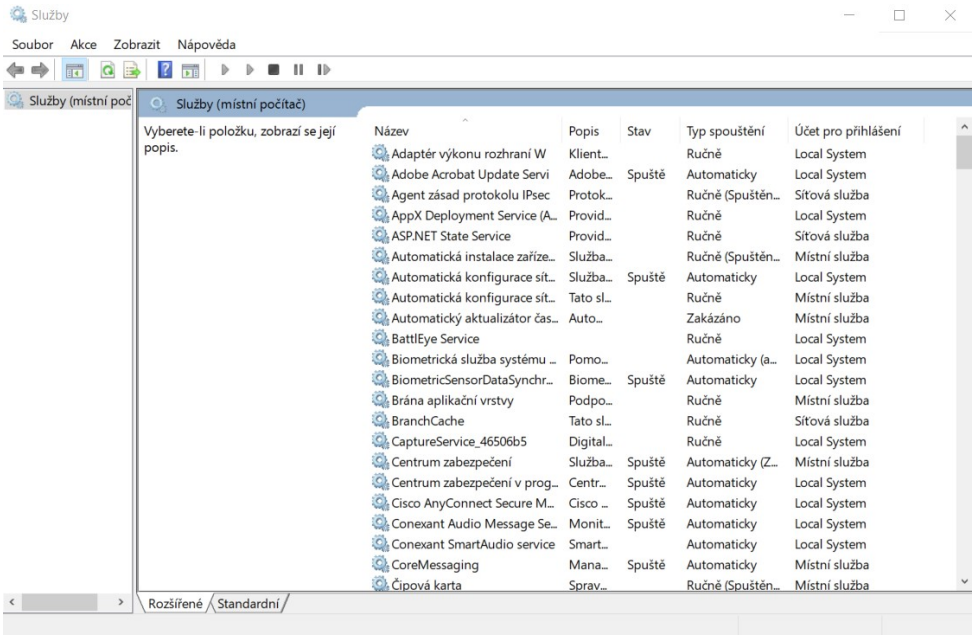
5.3 Služby

Windows služby jsou aplikace běžící na pozadí, které nemají uživatelské rozhraní. Jsou určeny primárně k poskytování informací o systému a k provádění pravidelných úloh.[7] Služby běžící na pozadí spuštěného systému mohou představovat riziko, neboť ne všechny služby vyvinula společnost Microsoft. Některé aplikace a ovladače instalují své služby. Služby mohou být spuštěny během bootování systému před ostatními programy a ještě před tím, než se uživatel vůbec přihlásí.

Uživatel by se měl zajímat o to, jaké služby na jeho počítači pracují a jaký je jejich účel. Je vhodné službu, jejíž funkcionalita není potřebná deaktivovat za účelem zrychlení systému.

K monitorování služeb lze opět využít integrovaný nástroj „Správce úloh“, kde záložka „Služby“ představuje miniaturní verzi konzole pro správu služeb. K této konzoli se ostatně dá dostat klepnutím na tlačítko „Otevřít okno Služby“.

Obrázek 5: Správce služeb windows



Na obrázku 5 můžeme vidět právě toto systémové okno sloužící ke správě služeb. Každá služba se může vyskytovat buď ve stavu „Zastaveno“ nebo ve stavu „Spuštěno“. Zobrazeným službám lze kromě změny stavu změnit také typ spuštění.

OS Windows uživatele neinformuje o nově nainstalovaných službách a změnách jejich stavů a vlastností. Balíček *ServiceProcess* a jeho třídy *ServiceServiceController* jazyka C# umožňuje monitorovat a spravovat služby.

5.4 Uložené soubory

Monitorováním změn v dané adresářové struktuře lze odhalit škodlivou činnost softwaru v ní pracujícího nebo jiného softwaru s úmyslem poškodit ten první.

Systém Windows defaultně nenabízí uživateli nástroj, kterým by zprostředkovával přehled změn v jím zvolené adresářové struktuře. Existuje však velké množství externích programů, které to umožňují, nebo lze také tuto funkcionalitu zajistit jednoduchým PowerShell skriptem.

5.5 Síť

Monitorování sítě se nejčastěji implementuje pro předcházení problémů s konektivitou způsobených přetíženými nebo havarovanými servery, síťovými připojeními nebo jinými zařízeními pro určení stavu webového serveru, apod.

Síťový provoz je však také jedním z nejčastějších přenašečů malware. Proto je důležité sledovat parametry sítě poskytující konektivitu a provoz mezi touto sítí a připojeným zařízením.

5.6 Externí média

Připojené externí médium může bez vědomí uživatele infikovat zařízení, na kterém pracuje, a to při přenosu škodlivé aplikace ukryté mezi přenášenými daty nebo i samovolně při provádění operace "Autorun".

Provádění Autorun metody by uživatel měl ve většině případů zakázat a spíše povolovat výjimku pro externí média, kterým důvěruje, a u kterých je provedení této metody naprostou nezbytností. Autorun lze zakázat v nastavení systému v záložce „Zařízení“ možnost „Automatické přehrávání“, kde je možné nastavit výchozí možnosti automatického přehrávání jak pro vyměnitelné jednotky, tak i pro paměťové karty.

5.7 Registry

Registr Windows je databáze konfiguračních nastavení nebo informací (např. ukládání systémových klíčů a hesel). Ve starších verzích systému Windows byla konfigurační nastavení uložena v souborech *.ini* (inicializace). Registr systému Windows byl dalším vývojovým krokem v konfiguračním úložišti systému Windows.

Nicméně mnoho programů, zejména přenosných programů, stále používá soubory *.ini* k ukládání konfiguračních informací. Logicky je registr rozdělen do několika složek, tzv. „hives“. Nesprávný zásah do registrů může způsobit poškození registru a fatální důsledky pro funkčnost samotných Windows.

Při instalaci nového softwaru se ve většině případů ukládají nové záznamy do Windows registru. Při monitorování pokusů o tyto zápisy je potenciálně možno odhalit instalaci nevyžádaného software. Avšak systém Windows neobsahuje nástroj, který by informoval uživatele o změnách v registrech.

6 Bezpečnostní mechanismy systému Windows

Jedním ze základních prvků zajišťující bezpečnost systému Windows je integrovaný program Windows Defender, který je byl blíže popsán v kapitole 3.5.8. Následující kapitola popisuje další bezpečností mechanismy v systému Windows.

6.1 Řízení uživatelských účtů

UAC (User Account Control) je bezpečnostní technologie, která zásadně zvyšuje bezpečnost celého systému.

Úkolem této technologie je zamezit spouštění takových procesů nebo zabránění takovým změnám, které by mohly mít za následek ohrožení systému. Může se například jednat o instalaci softwaru, spouštění aplikací, manipulaci se soubory v systémových složkách apod.

UAC si v potřebných chvílích vyžádá od uživatele potvrzení požadovaných uživatelských práv (administrátor) a pouze po úspěšné autorizaci uživatele provede zadanou akci.

Tento princip je velmi jednoduchý, ale i přes svůj nezpochybnitelný přínos se občas setkává s námitkami ze strany uživatelů, kteří nejsou spokojeni přerušováním jejich práce neustálými požadavky o potvrzení. Od systému Windows 7 lze UAC alespoň částečně konfigurovat, což vedlo k větší uživatelské přívětivosti.

6.2 Secured \ Measured Boot

Secured Boot je součástí specifikace UEFI (náhrada rozhraní BIOS), jehož cílem je zabránit malwaru (bootkit a rootkit) modifikovat proces zavádění operačního systému. Zaručuje, že zařízení spouští pouze software, kterému výrobce spouštěného zařízení důvěřuje. Po spuštění počítače se zkontroluje podpis každého bootovacího softwaru, včetně ovladačů firmwaru UEFI, aplikací EFI a operačního systému.

Measured Boot firmware zaznamená podrobná data o průběhu spouštění systému, Windows poté tato data ukládá do modulu Trusted Platform Module (TPM) a zpřístupní tento záznam důvěryhodnému serveru, který na jeho základě ověří průběh zavádění systému.

6.3 Omezený přístup k síti a zabezpečení sítě - Firewall

Firewall je software (případně i HW), který zabezpečuje a řídí síťový provoz a reguluje počty spojení. Jedná se o kontrolní bod či filtr, který určuje pravidla pro komunikaci mezi sítěmi. Jedná se prakticky o množinu pravidel, která říkají, jaké pakety mohou být v dané síťové úrovni propuštěny.

Nejednodušší možností pro rozhodnutí o potenciální škodlivosti paketů je identifikace cíle a zdroje dat, tedy identifikace zdrojových a cílových adres a portů. Dále se tato analýza opírá o informace o stavu spojení, znalost kontrolovaných protokolů, atd. Firewall prakticky představuje množinu pravidel, která říkají, jaké pakety propustit, a které naopak zahodit.

6.4 Šifrování disku - BitLocker

BitLocker je technologie šifrování pevných disků, která je k dispozici ve všech OS Windows od verze Vista výše. Umožňuje šifrování systémových i nesystémových oddílů popř. i vyměnitelná média.

Pro svou funkci BitLocker používá technologii TMP sloužící k ochraně klíčů používaných k zašifrování pevných disků a poskytuje ověření integrity při bootování systému. TPM poskytne šifrovací klíče pouze pro zavaděč operačního systému, pokud nejsou modifikovány jeho soubory. TPM není navrženo tak, aby odolávalo sofistikovaným hardwarovým útokům. Jakmile je operační systém spuštěn, nemá TPM žádnou ochranou funkci.[5]

6.5 Ochrana zdrojů systému - WRP

WRP (Windows Resource Protection) je nový název systému Windows File Protection (WFP). Tato technologie zabraňuje výměně základních systémových souborů, složek a klíčů registru, které jsou nainstalovány jako součást operačního systému.[6] Prostředky chráněné protokolem WRP lze měnit pouze pomocí nástrojů k tomu určených.

Aplikace by se neměly pokoušet o úpravu zdrojů chráněných protokolem WRP, protože jsou používány systémem Windows a dalšími aplikacemi. Ochrana těchto prostředků zabraňuje selhání aplikací a operačního systému.

7 Návrh konceptu

Mezi širokou škálou virů, které jsou známy, existují některé obzvláště komplikované a sofistikované. Namísto toho, aby přímo směřovali k cílům nejdříve neutralizují ochranu počítače tím, že zablokují antivirový program a poté se volně usadí v operačním systému.

Hlavní náplní této práce bylo přijít s konceptem ochrany antivirového řešení proti napadení. Tedy to co komerční řešení nejčastěji označují jako „self-defense“ mechanismy, tedy mechanismy samozabezpečení, kdy se samotný antivirus brání násilnému ukončení pomocí nejrůznějších prostředků. Skutečné postupy, kterými se dostupné antivirové nástroje ochraňují se mi dohledat nepodařilo. Přirozeně si firmy informace o použitých postupech pro zabezpečení svých produktů úzkostlivě střeží, a zveřejnění takovýchto citlivých informací by zcela jistě vedlo ke zranitelnosti samotných antivirů vůči útokům. Při návrhu konceptu ochrany AV jsem tedy vycházel ze svých domněnek, obecných principů zabezpečení aplikací v systému Windows a „násilného“ testování některých komerčních řešení spolu s průzkumem jejich integrace do systému, kde s použitím administrátorských oprávnění většinou došlo k omezení jejich činnosti do dalšího restartování systému. Metody, které jsem k tomuto účelu použil, jsem „obrátil“ a zahrnul do svého návrhu.

Navržená aplikace by tak měla implementovat mechanismy samozabezpečení s využitím monitorování systému a změn v něm jako použité a nově vytvářené procesy a služby.

7.1 Skener - chráněná aplikace

Mým hlavním úkolem, v této části práce, je prozkoumání možných technik a jejich aplikování na skutečnou aplikaci, kde cílem je vytvoření „nevypnutelné“ chráněné aplikace simulující tak koncept chráněného antiviru (dále jako „chráněná aplikace“ či „skener“). Zatímco běžné antivirové produkty implementují „self-defence“ mechanismy jejichž funkcí je spíše zabránit nevyžádanému poškození bezpečnostního produktu, ve svém konceptu pracuji s předpokladem že uživatel pracuje na svém zařízení jako administrátor a chce aplikaci „nekorektně“ vypnout. Tedy pro testovaní v rámci výuky kde by studenti byli vyzvání k instalaci a spuštění aplikace a k pokusu o její násilné vypnutí. Koncept si tak v podstatě klade za cíl větší zabezpečení, které by ho ochránilo i před pokročilejším retrovirem využívajícím technik pro eskalování svých oprávnění v systému.

Pro funkcionalitu chráněné aplikace je vyžadováno spuštění pod administrátorským účtem.

7.2 Samoobnovení spuštěných procesů

Nejběžnější metodou pro ochranu běžícího procesu před vypnutím je vytvoření druhého, paralelně běžícího procesu, přičemž pak tyto procesy opakovaně kontrolují existenci druhého. Pokud je jeden z nich ukončen, druhý ho obnoví. Touto jednoduchou kontrolou se velmi stíží jakýkoli pokus o nepodporované vypnutí aplikace

Správce úloh obsahuje možnost vynutit ukončení celého procesního stromu a ukončit tak daný proces a všechny jemu podřízené procesy. Pro druhý proces chránící skener se tedy hodí spíše použití služby běžící na pozadí.

7.3 Služba systému Windows chránící spuštěný skener

Předchozí podkapitola 7.2 počítá s vytvořením procesu, který běží paralelně se spuštěným skenerem, se kterým navzájem hlídají svou existenci v systému a v případě potřeby se spouštějí. Pro tuto funkcionalitu jsem zvolil službu systému Windows. Tato služba (dále i jako „podpůrná služba“) je tak instalována a spouštěna spolu se spuštěním chráněné aplikace v případě potřeby opakovaně.

Jako taková proto musí implementovat rozšiřující techniky pro spuštění chráněné aplikace, jako rozšíření možnost spuštění procesu obsahující GUI.

7.4 Monitorování systémových událostí

Zabezpečený antivirový systém musí monitorovat události v systému Windows a na jejich základě dále přizpůsobovat jak svou funkcionalitu, tak, v případě této práce, své zabezpečení.

7.4.1 Ochrana před potenciálně škodlivými procesy

Samoobnovení spuštěných procesů popsané v podkapitole 7.2 obsahuje zranitelnost v podobě prodlevy mezi jednotlivými kontrolami, které spuštěné procesy provádějí. Pokud jsou oba vypnuty ve stejnou chvíli dojde k násilnému ukončení aplikace. Služba tedy bude odchytávat událost systému značící spuštění nového procesu. A při každém spuštěném procesu vytvoří vlastní nový proces a to až do počtu 10 těchto nově běžících procesů. Tyto procesy pak kontrolují existenci obou entit, tedy jak chráněné aplikace tak služby a v případě potřeby je zase spustí. Každý proces před vykonáním kontroly obsahuje prodlevu v závislosti na pořadí, ve kterém byla daná instance spuštěna. Procesy tak postupně provedou svou činnost a ukončí se.

Tento proces se spouští s popisem „scvhost“, jedná se o záměrný překlep názvu „Svchost.exe“. Svchost.exe je jedním z nejznámějších systémových procesů, který je legitimní a bezpečný. Studenti pak při testování aplikace mohou tyto procesy zprvu přehlédnout.

7.4.2 Zachytávání událostí ukončení aplikace

Podobným způsobem jako v předchozí kapitole bude aplikace monitorovat i přímo své vlastní ukončení. Vznikne nová komponenta běžící na pozadí systému, která tak bude nezávislá jak na chráněné aplikaci tak na podpůrné službě. Tato aplikace bude odchytávat událost systému značící ukončení jakéhokoli procesu. Pokud ukončený proces bude představovat chráněný skener, aplikace jej znovu spustí s pomocí dostupných technik.

Samotný proces aplikace bude v systému opět vystupovat pod „skrytým“ názvem.

Alternativou znovuspuštění aplikace při jejím ukončení by mohlo být potlačení událostí pro ukončení procesů při tomto odchycení takovéto události, např. pomocí hookování Win API. Tato technika však nebude použita z důvodů popsaných v kapitole 7.10.3.

7.5 Zaznamenávání událostí

Zaznamenávání událostí (logování) je funkcionalita, kterou by správně navrhnutý software měl implementovat nejméně pro všechny své klíčové součásti. Výsledný log může při zpětné analýze mimojiné pomoci odhalit chyby vzniklé při běhu aplikace a přispět tak k její lepší funkcionalitě.

Jelikož cílem je vytvořit samozabezpečenou aplikaci, vznikne v tomto případě komponenta aplikace zachytávající stav chráněné aplikace a do přiloženého souboru bude zaznamenávat její ukončení a spuštění v čase. Výsledný log může být v tomto případě využit při testování aplikace pro usnadnění jejího ukončení.

7.6 Ochrana souborů aplikace

V případě běžící aplikace systém Windows sám ochrání její soubory před smazáním a to i v případě, že bychom se je nejprve pokusili přejmenovat a následně až odstranit. Tato ochrana však přirozeně pomine pokud aplikaci vypneme.

Samotný proces přejmenování, stejně jako přesunutí souboru v rámci diskové jednotky však systém Windows umožňuje. Některé z navrhnutých komponent budou pracovat s aktuálně spuštěným souborem (aplikace skeneru, služby, používané DLL, apod.), s jeho umístěním, ze kterého byl spuštěn a jeho názvem. Samotná implementace navrženého konceptu musí počítat i s tímto a ošetřit operace přesunu a přejmenování souboru tak, aby k těmto operacím nemohlo dojít.

V případě odinstalace řešení přirozeně k odstranění souboru dojde, což je žádaná skutečnost v systému pro jakékoli instalované řešení. Nicméně používané soubory jsou opět chráněny systémem a uživateli se tak zobrazí oznámení o nutnosti restartování systému pro dokončení odinstalace. Aplikace je tedy funkční právě až do restartu systému.

Skener pro svou funkcionalitu použije databázi uložených výsledků. Tato databáze se v případě smazání sama znovu vytvoří při další analýze. Obdobně bychom mohli obnovovat případné jiné soubory využívané aplikací nebo je v případě jejich nezbytnosti i zálohovat na různá místa na disku či dokonce online.

7.7 Naplánované samospuštění

Další vrstvu ochrany poskytuje naplánované samospuštění. Samospuštění bude provedeno pomocí vestavěného nástroje systému Windows „Plánovač úloh“ (Windows Task Scheduler[18]). Plánovač úloh umožňuje automaticky provádět rutinní úlohy v systému při splnění jakýchkoli nastavených kritérií. Tuto operaci pak provádí v předem určeném čase nebo opakovaně v nastaveném intervalu za nastavených podmínek.

Při startu aplikace je naplánovaná nová úloha s pravidelným a nekonečným intervalem. V každé iteraci tak zkontroluje existenci procesu chráněné aplikace v systému a v případě potřeby jej spustí. Tato úloha bude odstraněna a tedy přerušena jen v případě korektního ukončení aplikace.

Naplánovaná úloha ve své podstatě nikterak nesouvisí se samotnou aplikací a dojde-li jakýmkoli (nepodporovaným) způsobem k ukončení procesu skeneru, bude znovu spuštěn při další iteraci naplánované úlohy.

7.8 Automatické spouštění

Systém Windows umožňuje automatické spouštění aplikací při nebo po startu systému. Takovéto spouštění aplikací pomáhá správně inicializovat systém, šetří čas uživateli, který si tak může nechat své nejčastěji používané aplikace spouštět automaticky a v neposlední řadě se také automaticky spouštějí ochranné funkce a antivirové systémy, které tak mohou provádět kontrolu již se startem systému.

Ačkoli navrhovaná chráněná aplikace neprovádí žádnou kontrolu systému při jeho startu, její automatické spuštění by přesto mělo být implementováno z důvodu její vlastní ochrany.

Od Windows 7 a starších verzích systému Windows je k dispozici složka "Startup", která se nachází v *C:\Users\název účtu\AppData\Roaming\Microsoft\Windows\Start Menu\Programs*. Umístěním zástupce aplikace do této složky nastavíme její automatické spouštění pro daného uživatele.

Další možností automatického spuštění aplikace po startu systému je přidání záznamu v registru, konkrétně se pak jedná o klíč „*\Software\Microsoft\Windows\CurrentVersion\Run*“, kde jako hodnotu nastavíme název spouštěné aplikace a cestu k ní.

V případě služeb systému Windows umožňuje již jejich instalační konfigurace nastavení automatického spuštění, a to výběrem buď hodnoty „**Automatic**“, kdy se služba bude spouštět s bootovací sekvencí nebo „**Automatic (Delayed Start)**“, kdy se služba spustí až po spuštění systému.

Samotná funkcionalita automatického spuštění je zahrnuta převážně pro doplnění konceptu. Testující student by se měl zaměřit na násilné ukončení chráněné aplikace při jejím běhu bez restartu systému.

7.9 Poučený uživatel - pasivní zabezpečení

Sebelepší koncept zabezpečení antivirového softwaru neobstojí, pokud je (neúmyslně) „sabotován“ ze strany uživatele systému.

Útočník může využít metodik sociálního inženýrství[21] s cílem zmanipulovat uživatele tak aby sám zavedl virus do počítače za účelem překonání antivirové ochrany, nebo aby dokonce naivně

[21] Více o sociálním inženýrství v kapitole 2.2.7.

a dobrovolně ukončil antivirové procesy s použitím dostupných nástrojů a administrátorských práv.

Uživatelé by také z důvodu bezpečnosti neměli v systému pracovat pod účtem s administrátorskými právy pokud to nebude absolutně nezbytné. Jeho aktivita v systému pak nemůže být útočníkem tak snadno zneužita. Společnost vyvíjející komplexní antivirová řešení dostupná na trhu by měla dbát i na osvětu mezi uživateli.

Tyto prvky zabezpečení aplikace můžeme chápat jako metody pasivní bezpečnosti. Dále v této práci však implementovaný koncept nerozděluji na aktivní a pasivní zabezpečovací prvky, neboť koncept chápu jako celek a toto rozdělení by mohlo být zavádějící.

V případě této práce se přímo předpokládá aktivní snaha uživatele o nekorektní ukončení aplikace.

7.10 Další možnosti ochrany

Tato podkapitola obsahuje možnosti ochrany, které byly zvažovány, avšak nejsou zcela součástí hotového konceptu. Jedná se o metody, které by napomohly ochraně, ale nejsou použity z důvodu použití aplikace pro studijní účely, nebo zachování normální činnosti systému pro uživatele i při spuštěné aplikaci.

7.10.1 Náhodná jména všech komponent

Komponenty konceptu a jednotlivé instance těchto aplikací by mohly mít náhodně generované jména, či jména připomínající systémové prostředky pro zmatení útočníka a stížení vytvoření útočící aplikace. Tento postup by však v stejně tak studentům zabránil postupnému průzkumu aplikace a objevování jejích zabezpečovacích prvků. Podobně jako skrytí procesu ve správci úloh. Tato technika tak bude použita pouze u komponent popsaných v podkapitolách 7.4.1 a 7.4.2, tedy u konzolových aplikacích běžících na pozadí systému.

7.10.2 Potlačení nástrojů systému

Aplikace by byla schopna sledovat spuštění procesů představující nástroje systému Windows určené pro jeho správu, jako je například „Task Manager“, „RegEdit“, instance příkazové řádky apod. Vedlo by to však k potlačení funkčnosti systému a nejedná se o běžné chování antivirových řešení. Studentům by bylo stíženo intuitivní testování aplikace. Tyto techniky tak nebudou použity.

7.10.3 Potlačení ukončovacího signálu jednotlivých komponent

Služba systému Windows může být při své instalaci nastavena tak, aby nemohla být zastavena, pozastavena, či restartována. Toto nastavení však při implementaci nebude využito. Cílem je aby testující student mohl sledovat jisté „probliknutí“ při pokusu ukončit službu jakýmkoli

způsobem. Ze stejného důvodu nejsou použity ani techniky sloužící k potlačení ukončovacích událostí, které by mohly být vyvolány pro ukončení samotné aplikace.

7.10.4 Snížení intervalů

Většina navržených komponent pracuje při své funkcionalitě s nějakým intervalem. Tyto intervaly tak vesměs představují zranitelnost, neboť odhalují okno kdy je aplikace nehlídaná. Cílem opět je znázornění „probliknutí“ pro testujícího studenta a umožnění další práce. Tyto intervaly jsou tedy nastaveny tak, aby poskytovaly poměrně štědré okno příležitostí pro útočníka.

7.10.5 Násobné replikování použitých technik

Každá z navržených komponent pokrývá část zranitelnosti aplikace. Znásobením těchto komponent a jejich modifikací by bylo zajištěno větší míry ochrany. Mohlo by se jednat například o vytvoření několika dalších navzájem chránících se služeb, které by měly různé, náhodné jména, či by byly skryté jiným způsobem. Obdobně by se dala zreplikovat a zkombinovat většina použitých technik. Avšak pro použití pro studijní účely bude od této možnosti upuštěno.

8 Implementace navrženého konceptu

Tato kapitola popisuje konkrétní implementaci navržených komponent v kapitole 7. Veškeré zdrojové soubory jsou součástí přílohy A ve složce se zdrojovými soubory v podsložce s názvem „ScannerApp“.

Při implementaci jsem se rozhodl pracovat s knihovnou tříd WPF, která slouží primárně pro tvorbu grafického rozhraní a je součástí .NET frameworku jazyka C#.

Pracoval jsem s předpokladem, že uživatel pracuje v systému Windows 10 s účtem, který má pověření správce systému. Veškeré stěžejní funkce aplikace zasahující do systému jsou ošetřeny výjimkami (neposkytujícími žádnou formu logu či jiné informace pro uživatele), neboť se předpokládá, že uživatel (nebo potenciální retrovirus) bude pracovat s nejrůznějšími nekorektními metodami, aby docílil ukončení aplikace nebo některé její části. Je proto potlačen pád aplikace při ukončení jedné z jejích součástí, aniž by uživatel obdržel chybové oznámení, které by mu umožnilo „nahlédnout“ do zdrojových kódů.

8.1 Chráněná aplikace

Název projektu v řešení: ScannerApp

Pro samotnou simulaci antivirového programu (dále jako „chráněná aplikace“) jsem upravil aplikaci, kterou jsem vytvořil v rámci předmětu s názvem „Počítačové viry a bezpečnost počítačových systémů“, sloužící k odhalení malware v určené adresářové struktuře. Z důvodu kompatibility jsem zvolil adresář systému, který obsahuje dokumenty společné pro všechny uživatele tedy „*C:\Users\Public\Documents*“.

Změny ve sledovaném adresáři jsou odchytávány pomocí třídy „FileSystemWatcher“ nastavené tak, aby zachytila jakoukoli změnu ve všech podadresářích, tedy vytvoření, smazání či přejmenování souborů a složek, včetně provedených změn jako změna obsahu souboru, úprava oprávnění apod. Změněný či nově přidaný soubor do adresáře je pak automaticky podroben analýze.

Jedná se tedy o velmi jednoduchý skener, který pro svou činnost využívá API projektu virustotal.com[22]. Hodnota klíče pro toto API je uložena v konfiguračním souboru aplikace. Pro provolání tohoto API a analýzu souboru používám knihovnu „VirusTotalNET“ a HTTP požadavky na online verzi. Analýza jednoho souboru pomocí tohoto API se skládá se dvou požadavků. API umožňuje pouze 4 požadavky za minutu, je proto možné za minutu analyzovat pouze dva soubory. Mezi požadavky je také potřeba určitá prodleva, neboť analýza souboru chvíli trvá. Délka této prodlevy rovněž může být změněna v konfiguračním souboru aplikace (v odevzdaném řešení je prodleva nastavena na 5 sekund). Při kratší prodlevě je možné, že se uživatel nedobere výsledku analýzy, protože se aplikace doptává API dříve, nežli je výsledek k dispozici. Delší prodleva naopak může způsobovat zbytečné zpomalení aplikace. Aplikace ukládá výsledky

[22] https://developers.virustotal.com/v2.0/reference

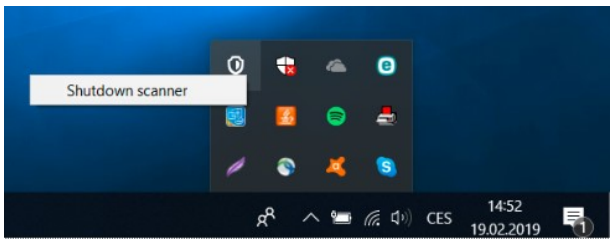
analýzy do lokální databáze ve formátu XML. Cesta k této databázi může být změněna v konfiguračním souboru aplikace. Pro každý analyzovaný soubor uloží nejen výsledek říkající, zda se jedná o virus a v případě, že ano tak jeho pravděpodobný typ, ale také hash obsahu souboru. Aplikace pro každý analyzovaný soubor nejprve projde tuto databázi a na základě hashe zjistí, jestli už byl soubor analyzován. Pokud najde shodný hash s hasem právě analyzovaného souboru, neposílá požadavek na API, ale rovnou uživateli zobrazí výsledek z databáze. Do aplikace se ukládají výsledky pouze korektně provedených analýz. V případě jakékoli chyby při analýze je tak dotyčný soubor analyzován znovu. Ukázku z databáze lze vidět na obrázku 6.

Obrázek 6: XML databáze chráněné aplikace shromažďující záznamy z analýzy souborů

```
<?xml version="1.0" encoding="utf-8"?>
<files>
  <file hash="e1f65fd2dd38e940d07bc82047867f60" isMalware="False" note="-" />
  <file hash="102dbbcdc1e7fea8f8c166bf2c81585b" isMalware="True" note="a variant of MSIL/Spy.Keylogger.CEP" />
</files>
```

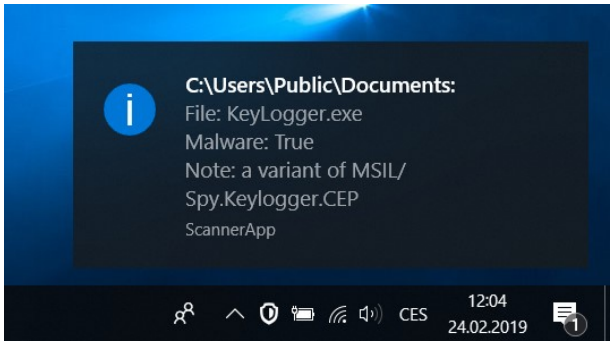
Aplikace neobsahuje žádné aktivní okno a pracuje „na pozadí“. Pomocí jednoduché vlastnosti WPF aplikací („ShowInTaskbar“) jsem tuto aplikaci velmi jednoduše skryl i v programu Task manager, kde je tak pouze k nalezení proces této aplikace nikoli aplikace samotná v příslušné sekci. Její aktivitu indikuje ikona zobrazená na hlavním panelu v oznamovací oblasti viz obrázek 7. Tato ikona představuje jediný korektní způsob vypnutí chráněné aplikace a to skrze akci v kontextovém menu, které lze zobrazit pomocí pravého tlačítka.

Obrázek 7: Ikona chráněné aplikace v oznamovací oblasti hlavního panelu



Zároveň jsou prostřednictvím této ikony přidávány a zobrazovány události notifikačního centra systému Windows zobrazující výsledek analýzy (viz obrázek 8).

Obrázek 8: Oznámení s výsledkem analýzy provedené chráněnou aplikací



8.2 Služba systému Windows chránící spuštěný skener

Název projektu v řešení: ScannerAppService

Jako komponentu chránící proces samotné aplikace a poskytující jeho znovuspouštění v případě nekorektního ukončení jsem zvolil službu systému Windows[23]. Jedná se tedy o projekt „Windows Service“, který se skládá z vlastní funkcionality služby, nástrojů pro řízení služby a také z konfiguračních a instalačních komponent umožňujících instalaci služby a zápis do registru.

Služba tedy opakovaně kontroluje existenci procesu chráněné aplikace v systému a v případě, že jej nenajde, pokusí se o jeho spuštění. Cesta k aplikaci představující skener je získána prostřednictvím záznamu v registru služby, neboť ten obsahuje záznam o místě odkud byla služba spuštěna.

Windows Service nemá grafické uživatelské rozhraní, protože by měla fungovat bez nutnosti jakéhokoli zásahu od uživatele a jsou zde spousty problémů spojených s nalezením a komunikací se správnou uživatelskou pracovní plochou. Přirozeně tak nemohla spustit chráněnou aplikaci vytvořenou jako projekt „Windows Presentation Foundation“. Cílem tedy bylo spustit interaktivní proces[24] z této služby s oprávněním správce. Pro tuto funkcionalitu je využita a upravena externí knihovna[25].

ScannerAppService také odchytává události systému značící spuštění nového procesu a následně spouští projekt „ScannerConsoleWatcher“ popsaný v kapitole 8.3.1.

8.3 Odchytávání událostí systému - WMI

Komponenty popsané v této kapitole využívají pro svou funkcionalitu rozhraní WMI (Windows Management Instrumentation).

WMI je rozhraní, které společnost Microsoft vytvořila pro poskytování informací z prostředí Windows. Infrastruktura WMI je založena na standardech WBEM (Web-Based Enterprise Management) a slouží k řízení operačního systému a shromažďování informací o něm.[21]

[23] Více o službách Windows 5.3
[24] Interaktivní proces je takový, který je schopen zobrazit uživatelské rozhraní na ploše.
[25] https://www.codeproject.com/Articles/35773/Subverting-Vista-UAC-in-Both-and-bit-Archite

Nástroj WMI je předinstalován ve všech operačních systémech Windows 2000 nebo novějších od společnosti Microsoft a může být stažena pro starší operační systémy, jako jsou Windows NT, 95 a 98. Může být využíván jazyky VBScript, PowerShell a .NET.

8.3.1 Aplikace spouštěna při inicializaci nového procesu

Název projektu v řešení: ScannerConsoleWatcher

Implementace komponenty popsané v kapitole 7.4.1 je zrealizována pomocí projektu „Console Application“ jazyka C# (.Net). Tato aplikace z podpůrné služby, kde je pomocí rozhraní WMI implementováno zachycení události, která představuje spuštění nového procesu v systému. Při svém spuštění aplikace využívá knihovny „System.Diagnostics“, „System.Configuration.Install“ a „System.ServiceProcess“.

V případě potřeby dojde ke spuštění, potažmo instalaci služby chránící skener, a tak i k obnovení ochrany samotného skeneru a případně k jeho znovuspuštění.

Odkladem spuštění v závislosti na počtu instancí této aplikace běžících v systému, je dosaženo průběžné kontroly chráněné aplikace v řádech až desítek vteřin při každém nově spuštěném procesu v systému.

8.3.2 Událostí ukončení chráněné aplikace a její znovuspuštění

Název projektu v řešení: ScannerConsoleRunner

Komponenty popsané v kapitole 7.4.2 je opět zrealizována pomocí projektu „Console Application“ jazyka C# (.Net). Tato aplikace je spuštěná se při startu podpůrné služby a jako taková běží na pozadí systému bez viditelného okna. V nástroji „Správce úloh“ je tak viditelný pouze její jediný proces, který je však uveden pod názvem „explorer“.

Pomocí rozhraní WMI je pak implementováno zachycení události ukončení jakéhokoli procesu v systému Windows. Při každém zachycení se pak ověřuje, zda byl ukončen právě proces chráněné aplikace, a pakliže ano, je znovu spuštěn. Ukázka z implementace této funkcionality je zobrazena ve výpisu zdrojového kódu 1. Implementace předchozí kapitoly 8.3.1 je pak velmi podobná.

```
...
_managementEventWatcherStop = new ManagementEventWatcher(new WqlEventQuery("
    SELECT * FROM Win32_ProcessStopTrace"));
_managementEventWatcherStop.EventArrived+= new EventArrivedEventHandler(
    ManagementEventWatcherStop_EventArrived);
_managementEventWatcherStop.Start();
...

static void ManagementEventWatcherStop_EventArrived(object sender,
    EventArrivedEventArgs e)
```

```
{
   if (e.NewEvent.Properties["ProcessName"].Value.ToString() == processName + "
       .exe")
      ...
        Start ScannerApp
      ...
}
```

Výpis 1: Zachycení události ukončení procesu chráněné aplikace pomocí WMI

8.4 Ochrana souborů aplikace

Na základě kapitoly 7.6 byly v rámci chráněné aplikace implementovány instance knihovny *FileSystemWatcher*, pomocí kterých dochází k zachytávání události „Renamed“ a „Deleted“ vzniklých v rámci složky(„aktuální složka“), ze které byla aplikace ve své momentální instanci spuštěna a dále je pak zachytávána a zpracovávána i událost „Created“ v rámci všech diskových oddílů.

V rámci odchytávání těchto událostí dochází k potlačení přejmenování či přesunu libovolného souboru z aktuální složky pomocí jeho okamžitého zpětného přesunu na původní místo, nebo na téže místo pouze s jiným názvem v případě, že proběhlo přejmenování. Ke zpracování události „Created“ dojde pouze v případě, je-li předtím zachycena událost „Deleted“ v aktualní složce.

Uživatel či potenciální retrovirus tak není schopen takto odstranit potřebné soubory. V případě pokusu o přejmenování souboru je uživateli zobrazena chybová hláška (pokus o přesunutí neexistujícího souboru). Při pokusu o přesunutí souboru pak může uživatel počítače se slabším hardware pozorovat „probliknutí“ souboru v dané složce než-li je přesunut zpět. Alternativou by mohlo být použití Windows API s hookováním těchto událostí a jejich potlačení.

8.5 Logování

Chráněná aplikace odchytává události z chráněné aplikace a zapisuje je do souboru s názvem „scanner_log“, který se ukládá do téže složky odkud byla aplikace spuštěna. Při přesunu aplikace se tak nepotlačí funkcionalita záznamu, je pouze vytvářen na novém místě.

Zaznamenává se ukončení a spuštění aplikace. V případě ukončení je zaznamenán čas. Při spuštění je kromě času uveden i proces, ze kterého byla aplikace spuštěna (tedy její „rodič“), což je zjištěno pomocí Windows API, kde framework .NET nabízí služby „Platform Invocation Services“ (PInvoke). Toto rozšíření obsažené ve jmenném prostoru „System.Runtime.InteropServices“ umožňuje volání nativních knihoven (*DllImport*). V našem případě je využita knihovna *kernel32.dll* pracující s jádrem operačního systému. Ukázku z implementace této funkcionality pro zjištění „rodiče“ procesu chráněné aplikace zobrazuje výpis 2.

```
...
```

```
[DllImport("kernel32.dll")]
static extern bool Process32Next(IntPtr hSnapshot, ref PROCESSENTRY32 lppe);
...
while (parrentID == 0 && Process32Next(oHnd, ref oProcInfo))
{
   if (Process.GetCurrentProcess().Id == oProcInfo.th32ProcessID)
   {
      parrentID = (int)oProcInfo.th32ParentProcessID;
   }
}
...
```

Výpis 2: Ukázka z metody identifikující proces spouštějící chráněnou aplikaci

Výsledný log tak může usnadnit testování aplikace a ve své podstatě se ve výsledku jedná spíše o ulehčení penetračního testování. Ukázku výsledného logu zobrazuje obrázek 12 v kapitole 9.2.

8.6 Naplánované samospuštění

Pro samospuštění chráněné aplikace v daném intervalu je dle návrhu v kapitole 7.7 vestavěného nástroje systému Windows, a sice „Plánovač úloh“ (Windows Task Scheduler).

Úloha je v Plánovači úloh vytvořena pomocí knihovny framaworku .NET „Task Scheduler Managed Wrapper“[26] tak, že je od svého zavedení pravidelně spouštěna každou minutu. Ukázku kódu z nastavení této úlohy a potažmo práce s uvedenou knihovnou je zobrazena ve výpisu kódu 3.

```
using (TaskService ts = new TaskService())
{
   TaskDefinition td = ts.NewTask();
   td.Principal.RunLevel = TaskRunLevel.Highest;
   td.RegistrationInfo.Description = "Re-run ScannerApp";

   DailyTrigger dt = new DailyTrigger();
   dt.Repetition.Interval = TimeSpan.FromMinutes(1);

   td.Triggers.Add(dt);
   td.Actions.Add(new ExecAction(path, null, directory));
   ts.RootFolder.RegisterTaskDefinition(@"ScannerApp", td);
}
```

[26] https://github.com/dahall/taskscheduler

Výpis 3: Nastavení úlohy pro samospuštění chráněné aplikace

Z výpisu 3 lze vyčíst, že úloha je kvůli své požadované funkcionalitě spouštěna s nejvyšším možným oprávněním.

Samotná aplikace díky své implementaci neumožňuje spuštění více než jedné vlastní instance. Nicméně spuštěná naplánovaná úloha nemůže být spuštěna znovu. Nedojde tak ke dvojímu spuštění chráněné aplikace.

Při korektním vypnutí aplikace prostřednictvím notifikační ikony je naplánovaná úloha z Task Scheduler odstraněna.

8.7 Automatické spouštění

Dle kapitoly 7.8 je také implementováno automatické spouštění chráněné aplikace při spuštění systému, a to dvojím způsobem pro zajištění vyšší úspěšnosti.

Při startu systému Windows dochází ke spuštění samotné chráněné aplikace, pakliže už byla v minulosti spuštěna. Při každém spuštění aplikace dojde k přidání (úpravě) záznamu v registru, v klíči „*\Software\Microsoft\Windows\CurrentVersion\Run*“, kde je jako hodnota nastavena cesta ke spouštěné aplikaci. Nastavení tohoto záznamu je provedeno kódem zobrazeným ve výpisu zdrojového kódu 4.

```
using (RegistryKey key = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\
    Windows\\CurrentVersion\\Run", true))
{
   key.SetValue("ScannerApp", "\"" + Assembly.GetExecutingAssembly().Location +
       "\"");
{
```

Výpis 4: Úprava registrů umožnující automatické spouštění aplikace při startu systému

Automaticky je spouštěná také služba systému chránící spuštěný skener, a to nastavením hodnoty „Automatic“ u položky „StartType“ v konfiguraci instalátoru služby. Služba tak bude spouštěna při bootování systému a po svém startu dle své funkčnosti také spustí samotnou chráněnou aplikaci.

Jak již bylo zmíněno, předpokládá se práce v systému jako administrátor. Testující student tak dle svých znalostí a zkušeností dokáže obě implementované metody automatického spuštění potlačit s pomocí nástrojů systému Windows. Důvod zachování této zranitelnosti je uveden v kapitole 7.10.2.

8.8 Instalátor kompletního řešení

Název projektu v řešení: ScannerAppAI

Pro usnadnění nasazení a šíření hotového řešení je vytvořen instalátor, který je součástí přílohy A v samostatné složce s názvem „Deploy“. A to pomocí nástroje „Advanced Installer“[27].

Průběh instalace je následně pro uživatele velmi jednoduchý. Po spuštění instalátoru je pouze vyzván k potvrzení instalace a výběru umístění instalované aplikace. Následně dokončí instalaci a aplikace je připravena k okamžitému spuštění a testování.

[27] https://marketplace.visualstudio.com/items?itemName=caphyon.AdvancedInstallerforVisualStudio2017

9 Testování

Tato kapitola rozšiřuje předchozí kapitolu 8 o další poznatky z průběžného i konečného testování implementovaného řešení. Aplikace je optimalizovaná pro použití v systému Windows 10. Při použití jiné verze operačního systému Windows nelze zcela zaručit bezchybné fungování všem implementovaných komponent.

V průběhu jsem pro testování používal a průběžně upravoval konzolovou aplikaci s názvem „AntiScannerApp“. Tato aplikace tak může být v případě potřeby použitá i pro nekorektní ukončení aplikace násilnou formou. Konečná verze této aplikace je součástí přílohy A ve složce se zdrojovými soubory v podsložce s názvem „AntiScannerApp“.

V průběhu implementace jsem řešení testoval průběžně. Veškeré komponenty popsány v kapitole 8 jsou zcela funkční dle návrhu v kapitole 7. Kapitoly 7 a 8 jsem v rámci testování dále rozšiřoval a nalezené nedostatky a další možnosti ochranz jsou tak již součástí hotového řešení, tedy navrhnutého a implementovaného konceptu ochrany. Tato kapitola dále popisuje průběh testování hotového řešení, kde samotný průběh testování může odpovídat předpokládané práci studenta, který dostane za úkol navrženou chráněnou aplikaci násilně ukončit.

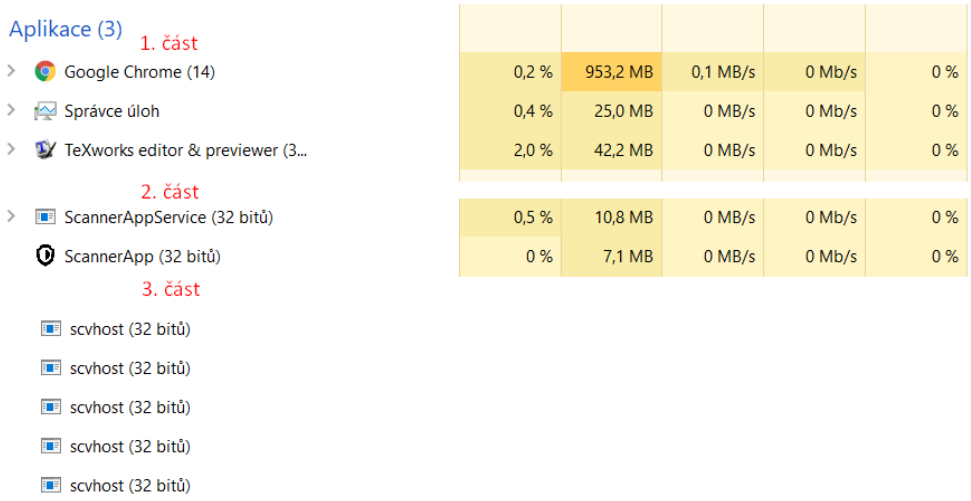
Aplikace může být v průběhu pokusu o její ukončení poznamenána tak, že ji nelze vypnout pomocí notifikační ikony. Tato situace může nastat, pokud je vypínáním poškozena některá z komponent, nebo pokud při opakovaném spuštění nebyla vykreslena notifikační ikona. V takovém případě je potřeba pokus o vypnutí opakovat, restartovat systém, opakovaně použít přiloženou anti-aplikaci nebo pokračovat v pokusech o jiný způsob vypnutí a donutit tak aplikaci k ukončení či znovuvykreslení ikony.

9.1 Správce úloh

Pokud chceme ukončit nějakou aplikaci, i méně zkušeného uživatele napadne ukončit tuto aplikaci pomocí integrovaného nástroje „Správce úloh“.

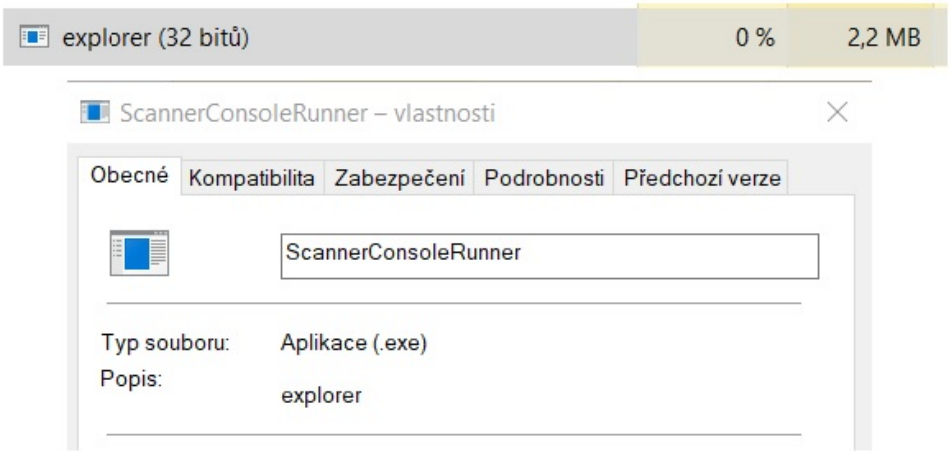
Na obrázku 9 vidíme veškeré informace související se spuštěnou aplikací, které jsou dostupné v záložce „Procesy“. V první části tohoto obrázku si lze všimnout, že chráněná aplikace není zobrazená mezi spuštěnými aplikacemi a je tak v první fázi před uživatelem skrytá. Vyhledáním procesů dle jména aplikace se přesuneme k části druhé, kde lze vidět proces spuštěné aplikace, ale také proces představující službu popsanou v kapitole 7.3. Tyto dva procesy se navzájem kontrolují a v případě potřeby se téměř ihned spouštějí. Jejich ukončení pomocí pravého tlačítka a možnosti „Ukončit úlohu“ tak není v podstatě možné (interval kontroly lze upravit v kódu aplikace). Ve třetí části vidíme skupinu procesů, které vycházejí z návrhu v kapitole 7.4.1. Tyto procesy se navzájem nechrání, avšak jejich upravený popis a vyšší počet instancí slouží ke zmatení „útočníka“ a jejich úkolem je postupně kontrolovat existenci jak chráněné aplikace tak podpůrné služby a v případě potřeby je spustit. Případný útočník tak musí vynutit ukončení všech těchto skupin procesů ve stejnou chvíli.

Obrázek 9: Spuštěné procesy chráněné aplikace dostupné prostřednictvím nástroje Správce úloh



Obrázek 10 ukazuje spuštěný proces aplikace ScannerConsoleRunner. Ten je popsán jako „explorer“. Po rozkliknutí vlastností procesu lze však vidět, že původní jméno zůstalo zachováno. Útočník musí procesy komponent popsaných v kapitolách 7.4.1 a 7.4.2 vypnout pomocí jejich skutečného jména (popř. ID).[28]

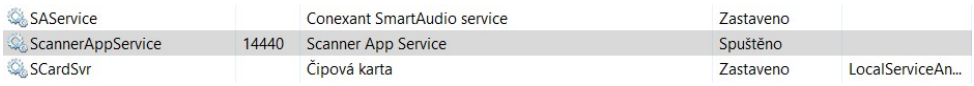
Obrázek 10: Spuštěný proces ScannerConsoleRunner s popisem „explorer“



Na obrázku 11 je zachycen prostor v záložce „Služby“, který obsahuje záznam podpůrné služby. Při pokusu o ukončení či odinstalování této služby pomocí dostupných možností je služba automaticky znovu spuštěna prostřednictvím chráněné aplikace.

[28] Při testování aplikace v systému Windows 7 se tyto spuštěné procesy ve většině případů zobrazovaly pod svým skutečným jménem a nastavené „krycí jméno“ (popis) bylo ignorováno.

Obrázek 11: Záznam podpůrné služby chráněné aplikace v nástroji Správce úloh

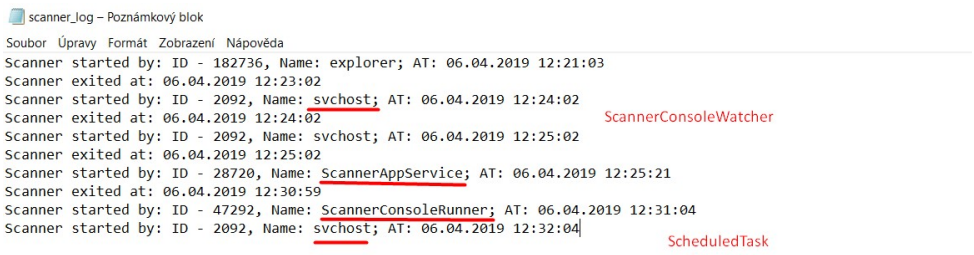


9.2 Logování

Pro testování zaznamenávání událostí spouštění a ukončování chráněné aplikace jsem pracoval s aplikací „AntiScannerApp“, pomocí které jsem vypínal některé ze zabezpečovacích komponent a „donutil“ tak skener spouštět se vždy pomocí jiné komponenty. Tímto způsobem jsem tak znovu ověřil funkčnost všech komponent a simuloval předpokládaný průběh penetračního testování.

Ukázku výsledného logu zobrazuje obrázek 12, na kterém můžeme vidět, že všechny implementované komponenty provádějí svou funkci. Komponenta „ScannerConsoleWatcher“ je velmi invazivní, k jejímu spuštění dojde i v případě spuštění naplánované úlohy. Také si můžeme povšimnout, že ne vždy dojde ke spuštění GUI v závislosti na tom, jak je komponenta spuštěna. Například spuštění pomocí naplánované úlohy nezobrazí GUI. Na funkci skeneru nemá chybějící GUI vliv, pouze nedojde k vykreslení notifikační ikony a tak nejsou vykreslována ani oznámení. Aplikace však soubory stále skenuje na pozadí a výsledky úspěšných analýz ukládá do databáze. V takovém případě lze stále najít běžící proces aplikace „ScannerApp“, který se snaží útočník vypnout.

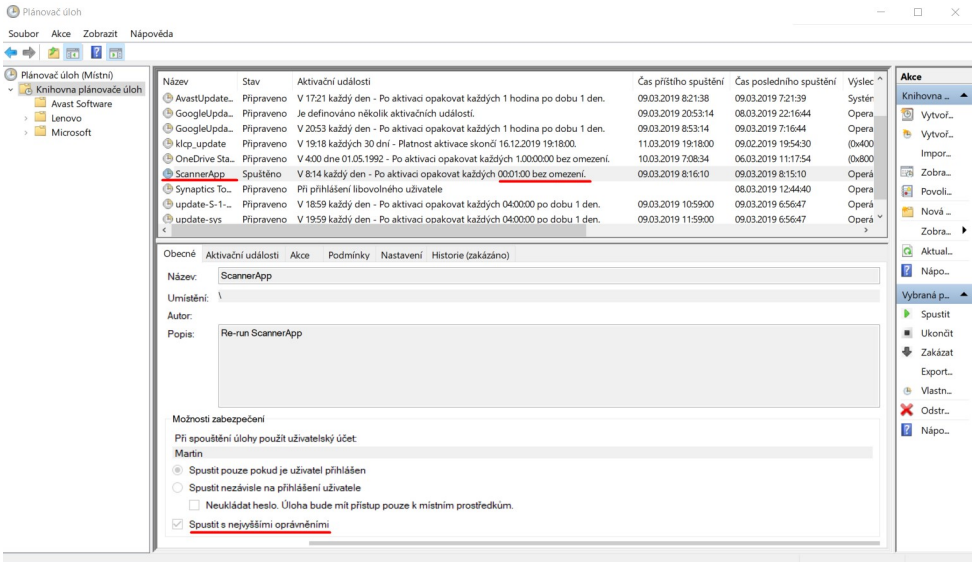
Obrázek 12: Ukázka souboru obsahující log stavu chráněné aplikace



9.3 Naplnánovaná úloha

Dojde-li k ukončení všech procesů zmíněných v předchozí kapitole, budou automaticky znovu spuštěny pomocí naplánované úlohy v Plánovači úloh (viz kapitola7.7). Tuto úlohu je tak možné v daném nástroji odstranit a následně ukončit procesy. Nastavená a připravená úloha je zobrazena na obrázku 13.

Obrázek 13: Záznam úlohy nástroje Windows Task Scheduler pro automatické spuštění chráněné aplikace



9.4 Automatické samospuštění

Automatické samospuštění aplikace po restartu systému popsané v kapitole 7.8 a 8.7 vykázalo při testování jisté neduhy. Aplikace se spouší bez GUI a nedojde tak k zobrazení notifikační ikony, tudíž je nemožné ukončit tuto aplikaci podporovaným způsobem do doby, než-li je aplikace znovu spuštěna jednou z metod zabezpečení. Nicméně funkčnost aplikace bude k dispozici i po restartu systému.

10 Závěr

Cílem této diplomové práce bylo seznámení se s postupy integrace antivirových programů do systému Windows a využití prostředků systému pro zabezpečení antivirového programu. Tedy návrh funkčního konceptu integrace antiviru do systému s ohledem na zabezpečení programu proti ochromení či zneškodnění.

V průběhu konzultací s vedoucím této práce bylo dohodnuto vytvoření konceptu a následná implementace „nevypnutelné“ aplikace, která by umožnila budoucím studentům našeho oboru pokusit se o její násilné ukončení a prozkoumat tak různé metodiky, které by případně mohly vést k omezení provozu jednoho z komerčních antivirových řešení dostupných na trhu.

Společnosti vyvíjející antivirové produkty si přirozeně informace technologických postupech použitých pro zabezpečení svých produktů chrání. Při zpracování teoretické části této práce jsem se pokusil některé firmy kontaktovat s žádostí o podklady, které by mi pomohly s popisem aktuální skutečnosti co se útoků vůči atvivirům týče, avšak pochopitelně bez jakékoli odezvy.

Při svém návrhu zabezpečení jsem tedy vycházel ze svých vlastních domněnek o zabezpečení aplikace v systému Windows a vlastních pokusů o násilné vypnutí či ochromení nejmenovaného komerčního řešení a průzkumu jeho integrace do systému. Inspirací mi také byly nejrůznější internetové diskuze zabývající se malware a jeho skrytím v systému a ochranou aplikací Windows proti jejich násilnému ukončení.

S ohledem na cíl práce jsem přizpůsobil koncept tak, abych studentům umožnil intuitivní testování této aplikace, aniž bych jim příliš usnadnil její násilné ukončení. Aplikace funguje s použitím administrátorských práv pro umožnění využití systémových prostředků. Rovněž se předpokládá, že i testující student bude při své práci přihlášen jako administrátor, neboť ukončení aplikace z režimu běžného uživatelského účtu považuji, z principu věci, za „teoreticky nemožné“. Koncept si tak v podstatě klade za cíl větší zabezpečení, které by ho ochránilo i před pokročilejším retrovirem využívajícím technik pro eskalovaní svých oprávnění v systému.

Celý navržený koncept je obsažen v kapitole 7. Její podkapitola 7.10 pak popisuje metody, které by napomohly ochraně, ale nejsou použity z důvodu použití aplikace pro studijní účely, nebo zachování normální činnosti systému pro uživatele i při spuštěné aplikaci. Navržený koncept nerozděluji na aktivní a pasivní zabezpečovací metodiky, neboť jej chápu jako celek.

Aplikace by studenty měla vést k postupnému odhalování jejich zabezpečovacích komponent, jejich prozkoumávání a testování. Cílem by mělo být objevení všech metod zabezpečení včetně detailních informací a následné úspěšné násilné vypnutí chráněné aplikace. Aplikace pro případ potřeby obsahuje i metody pro korektní ukončení. A v případě nutnosti je možno využít i přiloženou aplikaci sloužící k jejímu „násilnému“ ukončení.

Aplikace může být volně rozšiřována o další metody zabezpečení s použitím znalostí prostředí („frameworku“) .NET jazyka C#. Nicméně se jedná o funkční celek a věřím, že by mohl napomoci výuce jak teoretickými poznámkami, tak i s pomocí demonstrovaných ukázek.

Pro usnadnění nasazení a šíření hotového řešení je vytvořen instalátor, který je součástí přílohy A v samostatné složce s názvem „Deploy“.

Diplomovou prací hodnotím veskrze pozitivně, a to zejména z důvodu rozšíření mých znalostí z oblasti antivirových systémů. Jmenovitě pak získaný přehled při studiu možných technik zabezpečení a integrace antivirových software do systému Windows, je pro mě zajímavou a ceněnou znalostí. Také pak průzkum systému při návrhu konceptu a implementace navržených komponent pomocí jazyka C# jsou pro mě cennými zkušenostmi, které určitě v budoucnu využiji.

Literatura

[1] Joxean Koret, Elias Bachaalany, *The antivirus hacker's handbook*. Indianapolis, Indiana: Wiley, [2015]. ISBN 978-1-119-02875-8.Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein

[2] Statista – The portal for statistics, *Market share held by the leading Windows anti-malware application vendors worldwide, as of January 2018* [online]. [cit. 11.7.2018]. Dostupné z: https://www.statista.com/statistics/271048/market-share-held-by-antivirus-vendors-for-windows-systems/

[3] Microsoft Support, *What is a DLL?* [online]. [cit. 19.9.2018]. Dostupné z: https://support.microsoft.com/en-us/help/815065/what-is-a-dll

[4] Martin Dráb, *Jádro systému Windows: kompletní průvodce programátora.* Brno: Computer Press, 2011. Programování (Computer Press). ISBN 978-80-251-2731-5.

[5] Joel Scambray, Stuart McClure, *Hacking exposed Windows: Windows security secrets & solutions. 3rd ed.*. New York, NY: McGraw-Hill, c2008. ISBN 978-0071494267.

[6] Microsoft Docs, *About Windows Resource Protection* [online]. [cit. 18.7.2018]. Dostupné z: https://docs.microsoft.com/cs-cz/windows/security/information-protection/bitlocker/bitlocker-overview/

[7] DigitalCitizen, Codrut Neagu, *What are Windows services, what do they do and how do you manage them?* [online]. [cit. 16.10.2018]. Dostupné z: https://www.digitalcitizen.life/what-are-windows-services-what-they-do-how-manage-them

[8] Microsoft Docs, *User mode and kernel mode* [online]. [cit. 16.10.2018]. Dostupné z: https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/user-mode-and-kernel-mode

[9] MSDN Library, *Overview of the Windows API* [online]. [cit. 16.10.2018]. Dostupné z: https://msdn.microsoft.com/en-us/library/aa383723(VS.85).aspx

[10] PortableApps *Security* [online]. [cit. 4.11.2018]. Dostupné z: https://portableapps.com/apps/security

[11] Management Mania, *Sociální inženýrství (sociotechniky)* [online]. [cit. 4.11.2018]. Dostupné z: https://managementmania.com/cs/socialni-inzenyrstvi

[12] Zahra Bazrafshan ; Hashem Hashemi ; Seyed Mehdi Hazrati Fard ; Ali Hamzeh, *A survey on heuristic malware detection techniques.* The 5th Conference on Information and Knowledge Technology, [2013]. E-ISBN 978-1-4673-6490-4

[13] Info300.net, *Different Antivirus Software - Detection Methods* [online]. [cit. 13.11.2018]. Dostupné z: http://info300.net/alai2/Brief1.html

[14] TechTarget - SearchSecurity, Lenny Zeltser, *How antivirus software works: Virus detection techniques* [online]. [cit. 13.11.2018]. Dostupné z: https://searchsecurity.techtarget.com/tip/How-antivirus-software-works-Virus-detection-techniques

[15] BullGuard, *A definition of malware* [online]. [cit. 15.11.2018]. https://www.bullguard.com/bullguard-security-center/pc-security/computer-threats/malware-definition,-history-and-classification.aspx

[16] MalwareFox, Shawn Abraham *List of Types of Malware* [online]. [cit. 15.11.2018]. https://www.malwarefox.com/malware-types/

[17] Kaspersky Lab, Lenny Zeltser *What is Heuristic Analysis?* [online]. [cit. 8.12.2018]. Dostupné z: https://www.kaspersky.com/resource-center/definitions/heuristic-analysis

[18] Microsoft Docs *Task Scheduler* [online]. [cit. 9.3.2019]. Dostupné z: https://docs.microsoft.com/en-us/windows/desktop/taskschd/task-scheduler-start-page

[19] Microsoft Docs, Eliot Graff, Nathan Bazan *What is a driver?* [online]. [cit. 22.3.2019]. Dostupné z: https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/what-is-a-driver-

[20] Apriorit *Windows API Hooking* [online]. [cit. 22.3.2019]. Dostupné z: https://www.apriorit.com/dev-blog/160-apihooks

[21] Nathon Dalton *C# WMI TUTORIAL* [online]. [cit. 24.3.2019]. Dostupné z: https://nathondalton.wordpress.com/2015/02/03/c-wmi-tutorial/

A Příloha v elektronické formě

- **Deploy** - instalátor hotového řešení
- **TeX** - zdrojové soubory LaTeX pro vysázení tohoto textu včetně složky s použitými obrázky
- **Text_DP_FRY0050** - tento text diplomové práce ve formátu PDF
- **Sources_DP_FRY0050** - zdrojové soubory